MULTI-ROBOT COOPERATIVE SURVEILLANCE IN UNKNOWN ENVIRONMENTS

LIU ZHENG

NATIONAL UNIVERSITY OF SINGAPORE

2006

MULTI-ROBOT COOPERATIVE SURVEILLANCE IN UNKNOWN ENVIRONMENTS

LIU ZHENG

(B.Eng., Tsinghua University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2006

Acknowledgements

From admission to graduation, my years in National University of Singapore and Institute for Infocomm Research have been blessed and supported by many people.

First, I would like to thank my advisors, Professor Winston Khoon Guan Seah and Professor Marcelo H. Ang Jr., for their unwavering support, non-stop encouragement and unfailing patience throughout this research. Their inspiring guidance and abundant knowledge led me to the completion of this study.

I would also like to thank fellow students and staff in Control and Mechatronics Laboratory at National University of Singapore. I thank Tirthankar Bandyopadhyay, Terence Sit, Fei Wang and Jiayi Hu for their insightful suggestions and discussions.

I am grateful to my colleagues in TARANTULAS project at Institute for Infocomm Research. I am fortunate to have the chance to work with them: I thank Eddie Tan for his guidance in wireless networking; I thank Joo Ghee Lim for his support in hardware testing; I thank Hong Xiang for his help in simulation programming.

Special thanks go to my lab-mates: Junxia Zhang, Inn Inn Er, Choong Hock Mar, and Ricky Foo. I especially want to express my gratefulness to Hwee Xian Tan who spent countless effort on helping me conduct the simulations.

Last, but not least, I would like to express my deepest gratitude to my family and my girlfriend for the love and support. My parents extended their passion for studying to me, while my

grandparents were a constant source of support. I am grateful to my uncle for his encouragement and enthusiasm. I am also grateful to my girlfriend, for her patience and for helping me keep my life in proper perspective and balance.

Table of Contents

Summary

Surveillance is a broad research topic covering many aspects including exploration, target searching, target tracking, localization, etc. Traditional surveillance techniques rely on static sensory devices and centralized control architectures, such that the application is limited to in-door or urban areas, and the system is vulnerable. To improve surveillance performance, in recent years, intelligent mobile robots are applied to extend the coverage of environments, accelerate the searching of targets, enhance the performance of target tracking and monitoring, and increase the reliability of the system. How to design a high-performance, low-cost, and robust mobile-robot surveillance system has aroused great research interest.

This thesis presents a series of distributed multi-robot approaches for practical surveillance in unknown environments. The approaches cover exploration, target searching, target tracking, and localization problems. With respect to the exploration and target searching problems, distributed algorithms such as the potential field-based exploration, swarm intelligence exploration, landmark-based exploration, and hop-count gradient-oriented searching, are proposed (Seah et al., 2005, 2006). These methodologies can improve the observation of environments and shorten the searching time for targets. With respect to target tracking, an artificial potential field-based intelligent tracking algorithm is proposed to enable the cooperative behavior in tracking mobile targets (Liu et al., 2003, 2004a, 2004b). In addition, due to the complexity and uncertainty associated with tracking, two reinforcement learning-based algorithms are proposed (Liu et al., 2004c, 2005a, 2005b, 2006). These learning algorithms enable robots to learn the optimal strategy to track targets. With respect to the localization problem, an auction-based task allocation scheme is developed for a robot team to improve the hop-count-based localization, which is a simple and

scalable localization technique that can be widely applied to most real-world applications (Sit et al., 2007).

The proposed surveillance algorithms are tested using both simulations and real experiments as a part of TARANTULAS [1] (The All-teRrain Advanced NeTwork of Ubiquitous mobiLe Asynchronous Systems) project. The simulation is done in an integrated simulation environment that includes both robotics simulator (Player/Stage) and networking simulator (GloMoSim). The experiment is based on small-size robots (MRKIT) and middle-size robots (Koala) with wireless transceiver (MICAz). The obtained results demonstrate the efficacy of the proposed cooperative multi-robot surveillance systems.

[1] TARANTULAS project is funded under the A*STAR's Embedded Hybrid Systems Program, from year 2003 to 2006. A*STAR is the acronym of Agency for Science, Technology, and Research, Singapore.

List of Tables

List of Figures

1 INTRODUCTION

1.1 Multi-Robot Systems and Surveillance

Multi-robot systems have been extensively studied and applied in many research areas, such as cooperative material transportation, distributed sensing, exploration and mapping, team formation and marching, and robot soccer (Figure 1-1). These studies have remarkably improved the ability of the robots in accomplishing complex tasks, and thus have already had great impact on both research and industry.

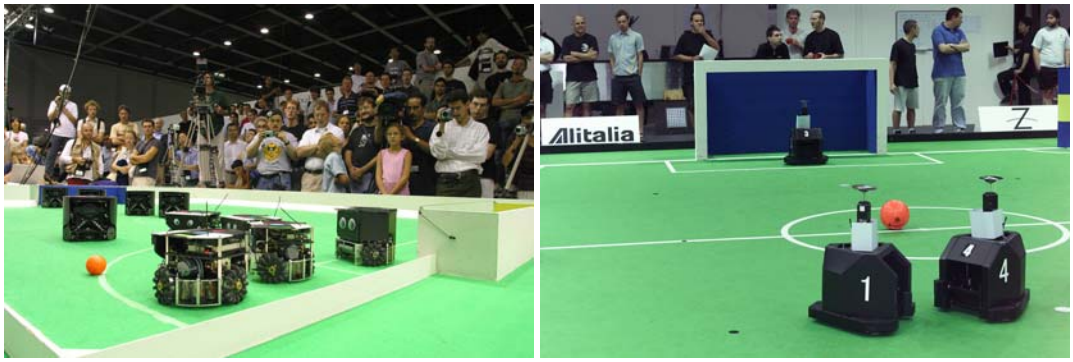

Figure 1-1 Robot Soccer Players (Photos from RobotCup Website)

In general, "surveillance" can be considered as "the act of carefully watching a person or place because they may be connected with criminal activities" (Longman, 1995). Surveillance systems are already widely used in our daily lives. From the perspective of robotics study, a complete surveillance system includes exploration and map building (Thrun, 2002), target searching (Ogras et al., 2004), target detection and identification (Reynaud & Puzenat, 2001), target tracking (Parker, 2002), localization (Olson, 2000), etc. The most basic study on surveillance is to find an optimal way of deploying stationary sensory devices to monitor the environment (Singer & Sea, 1973; Collins et al., 2001). With the advancement in technology, mobile robots/sensors are

introduced and applied to surveillance systems to extend the coverage of environments, accelerate the searching of targets, enhance the performance of target tracking and monitoring, and increase the reliability and robustness of the entire surveillance system.

Although multi-robot surveillance is superior in performance to single-robot systems, it requires appropriate control methodologies to achieve cooperation among mobile robots. Such control methodologies are usually complex and complicated, especially in the context of distributed control architecture. This motivates the study of multi-robot cooperative surveillance.

1.2 Research Motivation

As compared to single-robot systems, multi-robot systems are efficient, robust and economical (Cao et al., 1997). However, there are some significant challenges involved in the design of efficient multi-robot surveillance systems for real-life scenarios. This motivates the work presented in this thesis.

1.2.1 State of the Art: Surveillance

For many years, a vast range of surveillance algorithms have been proposed, studied and applied in both research and industry. In the context of robotics research, a surveillance system should include exploration and map building, target searching, target detection and identification, target tracking, target localization.

The initial surveillance systems are designed based on stationary sensory devices. The research purpose is to find the best position or pose for stationary sensors such that they can monitor the

environment efficiently, robustly and economically. Gradually, with advancements in technology, researchers are able to mount stationary sensors on mobile robots; therefore the performance of surveillance is greatly improved by the mobility of the robots. For example, a single sensor carried by a mobile robot can eventually obtain the information of the entire environment as the robot travels around. By this means, the single "mobile" sensor is equivalent to a large network of static sensors.

The primitive mobile robot surveillance system has been designed for a single robot or a small group of robots with low-level cooperation (Harmon, 1987; Rao & Iyengar, 1990). In recent years, increasingly cooperative multi-robot approaches have been proposed for surveillance (Parker 1997; Burgard et al., 2000; Zhang et al., 2005). These studies have shown the efficacy and robustness of the cooperative multi-robot systems; however, some of them require intensive computation power, powerful and accurate sensing ability, and reliable and rapid communications. Such high requirements may incur technical difficulties and excessive cost; therefore hamper the implementation in practical surveillance tasks. It is critical to design realistic systems that are reliable and achievable to real-life scenarios.

1.2.2 State of the Art: Multi-Robot Systems

In cooperative multi-robot systems, the essential research issue is to design and implement appropriate control methodologies to achieve cooperation among robots, taking into consideration of the various types of robots, tasks, and environments.

To achieve cooperation, many methods have been proposed and studied in recent years. The algorithms range from simple to complex, such as finite state automata (FSA), motivation-based behavior selection, the market-based contract method, etc. Normally, the simple control

methodologies (e.g., FSA) are practical and can be easily implemented, at the cost of low performance and cooperation level; on the other hand, the complex control methodologies (e.g., motivation/market-based methods) yield better performance, but are usually complicated and cannot be easily applied in real systems.

While it is difficult to achieve efficiency and applicability for general-purpose cooperative robot systems, it is even more difficult for surveillance tasks because they are usually executed in unknown environments. Without *a priori* knowledge, the uncertainties in the environment increase the complexity of system design. This problem has aroused great research interest in the study of robot learning that can let robots adapt to the environment and other variables to accomplish surveillance tasks.

1.2.3 Ideal Multi-Robot Surveillance

For both research and industrial applications, the desired multi-robot cooperative surveillance system should be efficient, feasible, scalable, and robust. The system should be able to achieve satisfactory performance with reasonable and affordable computation, sensing, communication, and other requirements. The system should be applicable in the real world with robustness to the errors in the system and the interference from the environment. The system should be able to work well with many different robots and sensors, and in a large-scale environment. Furthermore, the system should be reliable enough to continue working even if parts of the system, e.g., some robots, fail in functioning. This is the motivation of pursuing the study presented in this thesis.

1.3 Thesis Objectives

The aim of this study is to design an efficient, feasible, scalable and robust multi-robot surveillance system. This system should have the following basic functionalities:

- Ability to explore the environment and search for desired targets
- Ability to track moving targets for continuous and close observations
- Ability to localize the robots and other objects in the environment with acceptable accuracy

These functionalities are all in the context of multi-robot cooperative systems. With respect to exploration and target searching, the goal is to develop algorithms that can enable a group of robots to search in unknown environments to find targets with high efficiency and low costs in computation, sensing and communication.

In target tracking, the aim is to design control algorithms for multiple robots to track multiple moving targets cooperatively. The surveillance system should be able to assign targets to the most suitable robots. More importantly, such target selections should be done using distributed control methodologies to allow for scalability and robustness. In addition, since the movement of targets is usually unpredictable, it is highly desirable that machine learning methods can be applied to let robots learn how to cooperatively track targets without deliberative hardcoding.

With respect to localization, this study aims to find an efficient and scalable localization algorithm for a large multi-robot system in unknown environments. The localization algorithm should be able to provide accurate location information in real time because the robots and targets may move continuously; in addition, the localization algorithm should be applicable to simple

robots without reliance on expensive sensors or non-applicable sensors, e.g., Global Positioning Systems (GPS) which are not usable in indoor environments.

Since this study focuses on the control problems of mobile robots for practical surveillance in unknown environments, the following aspects of surveillance are beyond the scope of this thesis:

- Mathematical analysis of the surveillance system, such as the path planning problem or optimization problem. The proposed application environment is unknown, the task is complex, and the robots are cooperative, so it is not easy to model the system to analysis. Even if a simplified model may be used, it could not yield reliable results. Therefore, in this study, the system performance is tested and justified by realistic simulations and real robot tests.
- Map-building problem – the robots are not required to draw a map of the environment
- Target detection and identification problem – the robots are assumed to have the ability to recognize the targets from their sensor detections

1.4 Summary of Contributions

This thesis presents a series of distributed multi-robot approaches for surveillance. The purpose is to provide practical solutions for exploration, searching, tracking, and localization problems.

In most cases, the first step of surveillance is to find the targets (mobile/static, embodied/virtual) to observe or handle. This involves both exploration and target searching problems. In this thesis, three coverage-centric exploration algorithms, (i) potential field-based exploration; (ii) swarm intelligence exploration; and (iii) landmark-based exploration, are proposed to increase the coverage of the environment to help robots find targets. In addition, a target-centric search

algorithm, hop-count gradient-oriented searching, is introduced to enable robots to search for targets in promising areas by following useful clues. These exploration (coverage-centric) and searching (target-centric) algorithms (Seah et al., 2005, 2006) are effective in finding targets in large and unknown environments.

When the targets are found, it is important to track them for continuous observation. An artificial potential field-based intelligent tracking algorithm is proposed for this purpose. This algorithm allows cooperative robots to "select" suitable targets to track (i.e., solve the target assignment problem) according to their capabilities and feasibilities (Liu et al., 2003, 2004a, 2004b). This tracking algorithm is also distributed and highly scalable; therefore it can be applied to large robot teams. Because the targets are mobile and their mobility patterns are usually unknown, cooperative tracking is more difficult than exploration and target searching. To increase the adaptivity of cooperative robots and to avoid inflexible system design that is tailored for specific scenarios, two reinforcement learning based approaches, reinforcement learning in behavior-based control architectures and fuzzy reinforcement learning, are proposed (Liu et al., 2004c, 2005a, 2005b, 2006). These learning algorithms enable the robots to learn how to cooperate based on robot-robot and robot-environment interactions. In addition, a simple and distributed learning coordination scheme is developed to allow robots to learn concurrently with less interference.

With respect to the localization problem, a simple and scalable localization method based on the hop count is introduced. To address the intrinsic difficulty of hop-count-based localization, an auction-based task allocation scheme is proposed to enable multiple robots to improve the localization accuracy. Using very few robots, the localization accuracy can be remarkably improved (Sit et al., 2007).

In summary, the proposed multi-robot surveillance system covers almost all the aspects of conventional surveillance problems. Here is a list of the contributions of this thesis.

- Exploration and target searching
    - Improved searching performance. Compared to random search, the cooperative algorithms increase the average number of found targets, up to 20%, and reduced the average searching time, up to 30%.
    - Realistic requirements. The proposed exploration and searching algorithms are simple and scalable. The utilization of short-range-based communications reduces the interference among robots.
    - Reliable tests. The proposed exploration and searching algorithms are tested in simulations that integrate both robotic and communications simulators. The results are consistent to those of real robot systems.
- Target tracking
    - Improved tracking performance. In different scenarios, the robots can track more targets. The proposed tracking algorithms increase the average number of tracked targets, up to 10%.
    - Learning capabilities. Two learning algorithms are developed to allow multiple robots to choose the optimal strategies for tracking.
    - Realistic requirements. The proposed target-tracking algorithms are simple and scalable.
- Localization
    - Improved localization performance. Compared to random mobility, the proposed intelligent mobility of robots can help reduce the location error by 7%.
    - Realistic requirements. The proposed auction-based task allocation algorithms are simple and scalable.

In this thesis, the proposed distributed algorithms are simple and scalable such that they can be implemented in most real-world surveillance applications. Both simulations and experiments have shown the efficacy of the proposed multi-robot surveillance algorithms.

1.5 Organization of Thesis

In the following parts of this thesis, Chapter 2 introduces the related work in surveillance and multi-robot systems, including the main research issues and representative solutions. Following this, the proposed surveillance scenario and robot system are introduced in Chapter 3. Then, a series of surveillance algorithms are introduced in Chapters 4 to 6, namely the exploration and searching algorithms (Chapter 4), target tracking algorithms (Chapter 5), and the auction-based cooperation algorithm to enhance hop-count-based localization (Chapter 6). Finally, Chapter 7 concludes this thesis and suggests the direction of future research in multi-robot surveillance.

The organization of this thesis can be viewed pictorially in Figure 1-2.

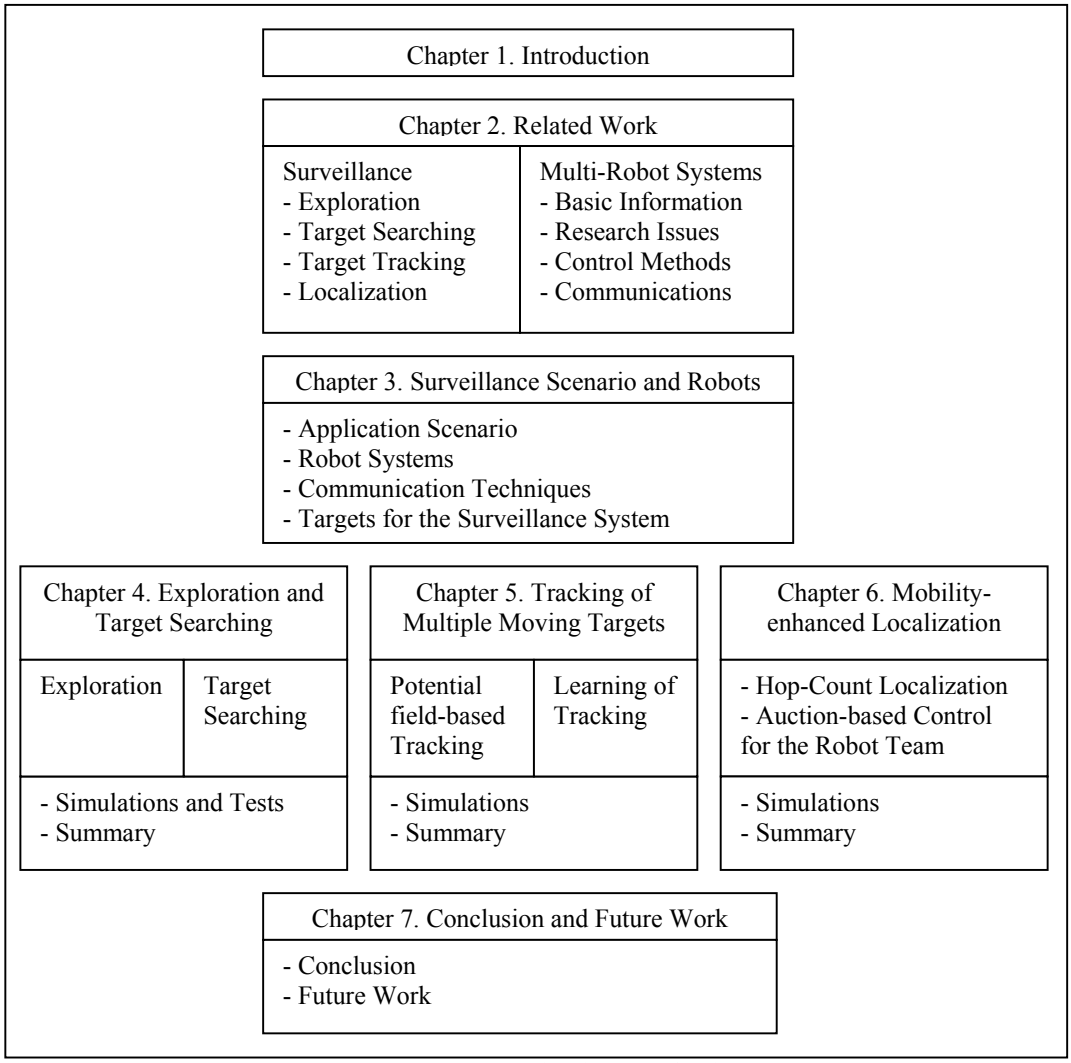

Figure 1-2 Organization of Thesis

2 RELATED WORK

The purpose of this study is to develop a series of control algorithms for multi-robot surveillance in unknown environments. This involves two large areas of robotic research – surveillance and cooperative robotics. In this chapter, a literature review on representative studies for surveillance and cooperative robotics is presented.

2.1 Surveillance

The research work for "surveillance" covers a series of research topics including environment exploration, target searching, target tracking, localization, etc. These topics can be categorized into the following areas:

- How to monitor the environment and find the targets to observe: exploration and target searching
- How to keep close/continuous observation of the targets (especially moving targets): target tracking
- How to obtain location information of the targets and robots: localization

In the following parts of this section, the related work in these areas is introduced and discussed, in the context of the following aspects:

- Requirements and assumptions
- Methodology and complexity
- Strengths and weaknesses
- Applicability
- Robustness

2.1.1 Exploration

The key problem for exploration can be explained as the following: “Given what you know about the world, where should you move to gain as much new information as possible?” (Yamauchi et al.,1999). A good exploration algorithm should have two properties, completeness and effectiveness. Completeness requires that the robot cover most of the environment; effectiveness emphasizes that the robot should achieve the completeness by minimal efforts, such as exploration time, exploration distance, etc.

Exploration algorithms are typically classified into two classes: deliberative exploration and reactive exploration. Deliberative exploration utilizes the map information of the environment to design the optimal routine or path to travel along. The map can be a complete global map that is known before the start of exploration, or a partial map that is built on-line while the robot explores.

In contrary to deliberative exploration, reactive exploration algorithms do not require the robot to hold or build a map. The exploration routine is decided according to the local or global (shared) observation of the environment. The temporal and spatial information may also be used to help in the reactive exploration.

2.1.1.1 Deliberative Exploration

In deliberative exploration, a map of the environment is required to aid the robot in finding the optimal motion trajectory to cover the entire region completely and effectively. If the complete map is known before the start of exploration, the designer of the robot system can decide the

optimal exploration routine for the robots. This is a path planning problem which is Non-Polynomial (NP) hard (Canny, 1988). To avoid the heavy computation in NP problems, some heuristics are proposed to find the sub-optimal solution for such path planning problems (Trevai et al., 2003).

In most realistic applications, the surveillance system designer does not have the complete map of the environment, and the environment is dynamic; therefore path planning cannot be applied. In this case, the most practical and applicable solution is to build the map while the robot explores, and use the map to guide the robot in further exploration of the unknown area. This methodology usually involves localization, and is known as Simultaneous Localization and Mapping (SLAM) (Thrun, 2002; Dissanayake et al., 2000). In SLAM, the relationship among exploration, map building, and localization is shown in Figure 2-1.

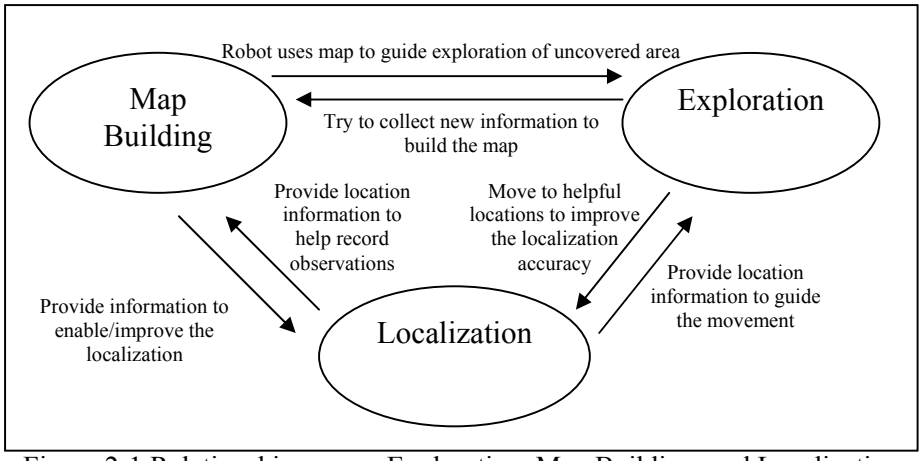

Figure 2-1 Relationship among Exploration, Map Building, and Localization

The maps used for deliberative exploration usually could be occupancy map, feature map, and topology (graph) map, ranging from simple to complex.

The occupancy map divides the entire region evenly into smaller grids, and records the occupancy information of each grid. The grid occupancy information can be in the form of discrete or continuous states (Thrun, 2002). For example, a grid may be in one of the following three discrete states: "-1" (empty), "0" (unknown), and "1" (occupied). In real exploration, the observation of the occupancy of a grid may not be accurate due to the uncertainties involved in sensing; therefore, the state of the grid state can also be represented using a continuous value between 0 and 1 to indicate the probability of occupancy status, e.g., 0.9 means the 90% probability that the grid is occupied. A grid map is simple and can be applied directly using sensor readings and location information. Based on occupancy maps, the most widely used deliberative exploration algorithm is the frontier-based exploration algorithm (Yamauchi, 1997). The basic idea is to identify the boundary of the covered area (frontier) in the map, and then select the appropriate frontier for the robot to move towards. By moving towards frontiers, the known area will increase continuously. For single-robot systems, there are two main research issues in this category of exploration algorithms:

- If there are multiple frontiers, which frontier should the robot move toward such that the information gain is maximized?
- How can the robot approach the frontiers efficiently and safely?

During exploration, the local map built by a robot may have more then one frontier to approach. In this case, the robot needs to estimate and compare the potential information gain of approaching each frontier and choose the best one. The potential information gain is the "utility" of the frontiers. According to different requirements or criteria, the calculation of the utility may vary. The simplest estimation of the utility is by calculating the distance from the robot to the frontier: the nearer frontier has higher utility (Yamauchi, 1997). In some applications, the calculation of the utility may give high priority to the regions of interest (Grabowski et al., 2003a),

pose separation (Grabowski et al., 2003b), important terrain property (Moorehead, 2002), certainty of the observation (Kobayashi et al., 2003), or traverse cost (Moorehead et al., 2001).

When the target frontier is assigned to a robot, the robot needs to approach this frontier using minimal moves while avoiding obstacles at the same time. This is a Non-Polynomial (NP) hard problem and can incur heavy computations in cluttered environments. To effectively plan the path to reach the frontier, probability-based algorithms, such as Probabilistic Road Map and Rapid-exploring Random Trees, can be applied (Calisi et al., 2005).

Comparing to single-robot systems, multi-robot systems involve additional research issues, as the following:

- For a team of robots, how can their local maps be merged to identify the global frontiers?
- For a team of robots, how can the frontiers be assigned to the most suitable robots?

To enable multiple robots to cooperatively explore the area using frontier-based exploration algorithms, the robots need to share their local maps to find the "global" frontiers. If the robots can perfectly localize themselves in the environment, they can share and merge their observations of the grids by simply adding or multiplying the state values in each local map (Yamauchi, 1998). However, if the localization information is not sufficiently accurate, the robots have to use probabilistic algorithms to merge the local map information. For example, particle filters can help a group of robots to merge local maps under the uncertainties associated with the robots' location estimates (Ko et al., 2003).

In multi-robot systems, the robots can negotiate to assign the frontiers to suitable robots when the local maps are merged into a global map. In Fang et al.'s approach (2004), a simple potential field-based algorithm allows each robot to choose the nearest frontier to approach. In addition, to

prevent a robot from trying to move to a near but unreachable target frontier (e.g., a nearby frontier behind the wall), higher priority is assigned to the visible frontier. A more optimal frontier assignment algorithm is to let the robots select frontiers in sequential order. When a robot has selected its destination frontier, the utility of this frontier will decrease so that the next robot will not choose this frontier again (Burgard et al., 2005).

Due to the simplicity of occupancy maps, frontier-based exploration has been widely used in both single-robot and multi-robot systems. However, the occupancy map has some intrinsic constraints that may impede the efficiency of the exploration. The following is a list of disadvantages associated with occupancy map-based deliberative exploration:

- The occupancy map usually increases proportionally with the size of the environment. If the environment is huge, the map will inevitably require large storage space.
- It is difficult to merge occupancy maps because the matching of grids may incur heavy computations.
- The resolution of the occupancy map is usually fixed because the grid size is fixed. It is hard to adaptively change the resolution according to task requirements.

To overcome the limitations of using occupancy maps, feature map-based deliberative exploration is proposed and studied. In contrast to the occupancy map, which stores all information regarding each grid, the feature map only records the existence of special features in the environment, e.g., corners, doors, walls, etc. (Thrun, 2002). The storage size of a feature map is only proportional to the number of features. Therefore, the feature maps usually take less memory space, and their merging is also simpler than that of occupancy maps. Similar to occupancy map-based exploration, the main objective of feature map-based exploration is to lead the robots to uncovered areas (Bauer & Rencken, 1995; Newman et al., 2003).

In addition to feature maps, topology map can also be used to help the robot in exploration. The topology map is a high-level map that only stores the connectivity (topological) information of different regions, e.g., room 1 is connected to room 2, but room 1 is not connected to room 3 (Thrun, 2002). By utilizing information obtained from topology maps, the robots can select uncovered regions to explore and search for targets (Wullschleger et al., 1999; Nagatani & Choset, 1999).

Regardless of what type of map is used, the common objective of deliberative exploration is to utilize the map information to find the optimal trajectory to move to the unknown area, with minimum re-exploration of known places. Such exploration algorithms can usually guarantee the full coverage of the entire environment, and their efficacy has been demonstrated by many real applications. However, deliberative exploration methodologies have some disadvantages, such as the following:

- Mapping of the environment can incur excessive computation and take up large storage space. If the environment is large with complex features, the robot needs to have powerful microcontrollers and large memory space.
- For multi-robot scenarios, cooperative exploration requires robots to share their local maps, which can incur intense information exchanges. Furthermore, since each robot may have some errors in its observation, probabilistic based filters, e.g., the Kalman filter (Kalman, 1960) or the particle filters (Metropolis & Ulam, 1949), have to be used to "merge" the local maps. This further increases the cost in computation and storage space.
- If the environment is dynamic, map building may not be helpful because the information in the map may not be accurate or reliable as time passes.
- If the environment is complex, e.g., an outdoor area, the mapping can be quite difficult and may hardly be accurate enough.

- Map building requires accurate and real-time location information; however, it is difficult to enable a group of simple robots to obtain localization information to a sufficient degree of accuracy.

Due to the above limitations, deliberative exploration is mostly applied in structured indoor environments using only a small team of powerful robots.

2.1.1.2 Reactive Exploration

In contrast to deliberative exploration, reactive exploration algorithms do not need the map information. The basic idea is to find the optimal move based on the current status of robots. For example, if a door is detected, the robot can move towards the door because it is a hint of an unknown area.

The most representative reactive exploration is the Artificial Potential Field (APF)-based approach (Khatib & Le Maitre, 1978). This approach considers the obstacles inside the environment because they are usually the key factors affecting the observation. For example, Howard et al. (2002) introduce an artificial potential field-based exploration algorithm for a team of robots. The main idea behind their approach is to map the sensed obstacles and robots as repulsive force sources and to allow the robots to move under such forces; therefore the robots can disperse within the whole area. The shortcoming of primitive potential field-based exploration is that there is no proactive driving force for the robots to move. If no obstacle or neighboring robot is detected, the robot will remain in its position with no intention of movement. To solve this problem, a random attractive force source is usually "given" to the robot to trigger its movement, i.e., the random exploration with obstacle avoidance.

A more sophisticated method of providing the driving force for robot exploration is the well-known visibility-based exploration. In this algorithm, the view area is generated using vision sensors or laser scanners. The robot can then identify and examine the boundaries of its visible regions to select the best orientation to move along (Huang & Gupta, 2005; Bandyopadhyay et al., 2005). This is known as the Next Best View (NBV) method.

For some special application environments, the robots may explore following the isoline in a virtual "potential field" (Baronov & Baillieul, 2007). The idea is to let the robots move along the curve where the potential value (e.g., radiation) is within a certain range, so that the robots can locate the source of the potential field. Usually this kind of exploration algorithms is limited to certain tasks with a source (e.g., radioactive material).

To maximize the coverage of the environment, the robots may explore in a certain formation, e.g., chain (Rogge & Aeyels, 2007). The idea is to let the robots maintain constant distance and angle while moving in a group. This kind of approaches is suitable for open areas without large concave obstacles; however, if the environment is complex, it is quite difficult for the robots to keep line-of-sight communications to maintain the formation.

For ad hoc sensor networks, Sugiyama et al. (2008) propose the algorithm that enables autonomous classification of robots' roles in exploration, by the forwarding table of each robot constructed for ad hoc networking. The main research idea is to maintain network communication, while exploring the environment.

In a sensor network, the robots may obtain useful information from static sensors for better exploration (Batalin & Sukhatme, 2007). The static sensors can store the data of Least Recently

Visited (LRV) as a clue for mobile robots to decide exploration direction. A topology graph of the environment is constructed by the static sensors in this study.

In comparison to deliberative exploration, reactive exploration is simple because it does not need the complex process of map building. It can perform well in complex environments and with a large number of robots. However, an apparent disadvantage of the reactive exploration is that complete coverage of the environment cannot be guaranteed. This is because the robot cannot remember the covered area without using a map.

2.1.2 Target Searching

In static environments, if the targets do not move, the target-searching problem is quite similar to the exploration problem. When the robots cover more regions in the environment, they are more likely to find the targets. In particular, if the robots are able to achieve full coverage, they can definitely find all the targets. In the literature, the random search is most commonly applied in target searching (Cheng & Leng, 2004). However, for multi-robot systems, it is important to coordinate robots to search efficiently. For example, by forming some special pattern and then marching in such pattern, the robots may have better sensing coverage and avoid the re-exploration of the area that has already been covered (Ogras et al., 2004).

If the targets are mobile, the search strategy is different. For example, if the targets are evasive and try to hide from the robots (searchers), they may reactively move to the blind regions of robots to avoid being found. In this case, even if the robots are able to achieve full coverage, they may not be able to find all the targets. A representative work for this category of target searching is to obtain a full-coverage exploration, with the consideration of eliminating blind regions during the exploration. To achieve this, the single-robot system (Suzuki & Yamashita, 1992) or multi-

robot system (Yamashita et al., 2001) should be controlled carefully to move along deliberatively scheduled routines.

In the real world, mobile targets may have a preference for some special regions in the environment. Therefore, it is reasonable to let the robots identify such special regions and assign higher priorities to search around them (Liu et al., 2004a). In addition, the robots can follow some target-related clues to search. For example, if the robots are able to identify the trail of targets, they can follow the trails to search for the targets.

In summary, the majority of related work simplifies the target-searching process in exploration and coverage problems; however, for mobile targets or intelligent targets, searching can be improved by considering and utilizing target-related information. This is one of the research problems studied in this thesis.

2.1.3 Target Tracking

Target tracking is one of the most important applications for security and surveillance. In this thesis, the "tracking" problem refers to the motion strategy for multiple robots to follow the targets to keep them within a certain range; however, virtual tracking is not considered such as identifying targets using visual data (Ito & Sakane, 2001; Schulz et al., 2001), sound data (Mattos & Grant, 2004), or other data sources.

In most applications, the objective of tracking is to keep the line-of-sight contact between robot and target within a certain range, e.g., by ensuring that the target can be constantly sensed by the laser scanner. Therefore, it is important to know the mobility of targets and the configuration of the environment, so that the robot is able to follow the movement of targets even in an

environment with many obstacles. If the mobility of targets is known or predictable (deterministic), robots can find the optimal strategy to track targets. This is usually a Non-Polynomial (NP) hard problem. For example, if the trajectories of targets are known, the robots can calculate the routes to follow targets in an offline manner (Efrat et al., 2003).

If the mobility of targets is unpredictable, the robots have to plan their motion in reaction to the motion of targets, in an online manner. In this case, the motion decision is usually generated based on the consideration that the targets should not be lost (Coue & Bessiere, 2001; Gonzalez-Banos et al., 2002; Murrieta et al., 2004; Muppirala et al., 2005).

In multi-robot scenarios, group cooperation may be used to track or capture targets. The cooperation can be generated by game theory (Skrzypczyk, 2004), social negotiation control (Krishna & Hexmoor, 2003), or region-based robot deployment (Jung & Sukhatme, 2002). In these approaches, each robot is assigned to track the most suitable target. However, most of these approaches require intense computation and explicit communications among members of the team, and are thus not scalable to large robot teams. This motivates the study presented in this thesis.

2.1.4 Localization

In robotics research, localization refers to the means of obtaining location information of robots or target objects, with respect to the environment features (relative localization) or the global coordinate system (absolute localization). Localization is necessary for many kinds of applications, such as exploration and map building, distributed sensing, pattern formation and following, robot soccer, etc.

For robots, one kind of localization methodology is by scan matching and global localization. The basic idea is to let the robot match the sensed data with previously sensed data (Burguera et al., 2007) or global map (Guivant & Katz, 2007); therefore it can estimate its location. Such localization methods have been widely used. However, due to the following constraints, it is not applicable for the proposed application scenario in this thesis:

- In this study, it is assumed that a global map is not available. This is to be introduced in Chapter 3.
- The simple and small robots may not have enough computation power or storage space to execute the scan matching algorithms.
- Due to the limitation of sensor quality, the noise in sensed data may badly degrade the performance for scan matching.
- For multi-robot systems, the scan matching of two robots' sensor data is challenging, especially if the robots are heterogeneous in sensing capability.

Another kind of localization methodology is triangulation. To apply this method, the following information is usually required:

- Reference point(s) – objects (either embodied or virtual) placed at known locations, e.g., satellites in space, wireless beacons, North Star, start point, a special land feature in the environment, etc.
- Measurement – information related to the reference point(s) and the robots (or receivers), e.g., TDOA (Time Difference of Arrival) of the signals from the satellites, RSSI (Received Signal Strength Index) from the wireless beacon, angle to North Star, distance and orientation to the start point, relative location to the land feature, etc.

By making use of the two kinds of information mentioned above, the robots can obtain the location information by performing some calculations. The simplest methodology is triangulation or multilateration (Langendoen & Reijers, 2003). If the locations of reference points and the distances to these points are known, triangulation can be used to estimate the location of the robots by minimizing the square errors of estimates, as shown in Equation (2.1). In this equation, (x, y) is the estimation of the location of robot; (x_i, y_i) is the location of reference point i; and d_i is the distance between robot and reference point i.

$$\text{Triangulation: } (x, y) \leftarrow \arg\min_{(x,y)} \sum_i (\sqrt{(x_i - x)^2 + (y_i - y)^2} - d_i)^2 \qquad (2.1)$$

To do triangulation, the distances or angles between the robot (or receiver) and reference points are indispensable information. For example, the Global Positioning System (GPS) needs to know the distances between the receiver and the satellites to locate the receiver. Usually, such distance information can be estimated directly by RSSI (Received Signal Strength Index), TOA (Time of Arrival), TDOA (Time Difference of Arrival), etc. In addition to distance measurements, the AOA (Angle of Arrival) from the reference to the receiver can also be used to estimate the location information.

For a large team of robots in a large environment, it is difficult to estimate the distances between robots and reference points accurately. To solve this problem, Approx Point-in-Triangulation (APIT) is proposed to reduce the reliance on the information of the distances between neighbors (He et al., 2003). However, this approach still makes use of distance information in that it requires knowledge of the changes in distance between objects in the environment.

Because all real-world sensors have errors and biases, it is desirable to utilize multiple sensors to get better location estimations because the sensors may cancel out the error estimates for each other. The cooperation of multiple robots can further reduce error estimates by mutual "calibration". The information obtained from different robots or sensors, however, can hardly be handled by triangulation alone. To solve this problem, some mechanisms have been proposed and applied to merge the multi-sensory information or multi-robot observations, such as the Kalman filter (Kalman, 1960) and the particle filter (Metropolis & Ulam, 1949). These probabilistic approaches require the robot to estimate its location, perform some actions and estimate the new states, observe the changes, and then adjust the estimation according to the observations. These approaches require complex computation (in sensor data processing and matching). Furthermore, the location estimation may take a long time to "converge" to the real location. If the robot team is large or the environment is complex, the state space (for the Kalman filter) or the number of particles (for the particle filter) will be excessively large and the whole system may not work appropriately. Therefore, the application of such approaches is normally limited to small robot groups or simple environments.

For the localization of large numbers of robots in big unknown environments, hop-count-based localization is practical due to its scalability. In this algorithm, hop-count information is used to estimate the distances between the robots and reference points, and triangulation is then applied to get the location of the robots (Langendoen & Reijers, 2003). Due to its simplicity and practicality, this localization algorithm has been widely implemented in many applications. However, there are some intrinsic limitations of this technique. One of the objectives of this thesis is to overcome the limitations of the hop-count-based localization to enhance its performance.

2.1.5 Summary

For the purpose of surveillance, the robot is required to explore the environment to search for targets, track targets for continuous observation, and obtain the location information of robots and targets.

In this subsection, the two exploration algorithms, deliberative exploration and reactive exploration, are introduced. Deliberative exploration is efficient because it usually can guarantee full coverage of the environment; however, the map-building process requires heavy computation and large memory storage, thus it is not very suitable to complex environments and large robot groups. On the other hand, reactive exploration is simple and scalable; but it can hardly guarantee the full coverage. This drives the motivation to further develop reactive exploration methodologies to achieve satisfactory exploration results that are comparable to deliberative exploration techniques.

In static environments, the search for stationary targets is similar to the exploration problem. However, if the environment is dynamic and the target is mobile and intelligent, searching becomes more challenging. It is important to utilize target-related information to improve the searching of targets.

Once the mobile targets are found, the robots should track them by keeping them under continuous observation. However, if the mobility of targets is unknown, the targets may be easily lost. Moreover, it is also difficult to assign suitable targets to the robots when there are multiple robots and targets. While the potential field-based tracking algorithm is widely applied due to its simplicity and scalability, it can hardly achieve the desired level of cooperation among the robots.

Therefore, better and more efficient algorithms should be developed to achieve a high-level cooperation among the robots for better tracking of mobile targets.

Localization is one of the most important issues in surveillance applications. It can provide the useful location information of targets, robots, or environments. In many applications, if the location of the target (or robot) is unknown, the measurements taken by the robot (e.g., temperature) are meaningless. There are many existing localization algorithms in the literature, e.g., range-based localization (triangulation), range-free localization and connectivity-based localization. Most of them can provide accurate location information; however, they have high requirements in computation power, memory space and sensor capabilities. They are not very applicable or scalable for large robot groups and complex environments. This motivates the study of simpler, more scalable and more cost-efficient localization techniques.

2.2 Cooperative Multi-Robot Systems

2.2.1 Definition and Classification

Cooperative multi-robot system research has attracted much interest in the last two decades. It includes a wide range of applications, such as multi-robot cooperative material transportation (Kube & Zhang, 1996; Miyata et al., 2002; Yamashita et al., 2003), distributed sensing (Parker, 2002; Jung & Sukhatme, 2002; Liu et al., 2004a), exploration and mapping (Grabowski & Kholsa, 2001; Thrun, 2002; Roumeliotis & Rekleitis 2003), team formation and convoying (Balch & Hybinette, 2000; Molnar & Starke, 2001; Fredslund & Mataric, 2002), and robot soccer (Kim & Vadakkepat, 2000; Bjorklund, 2002; Stone, 2003). The cooperative multi-robot system is more than just a simple extension of the single-robot system. It not only increases the performance and

robustness of the system by parallel operation; but is also able to accomplish tasks impossible to a single-robot system through "cooperation".

While the term "cooperation" is widely used in robotics research, very few papers define it explicitly. In the review paper by Cao et al (1997), "cooperation" is explained as the following: "Given some task specified by a designer, a multiple-robot system displays cooperative behavior if, due to some underlying mechanism (i.e., the "mechanism of cooperation"), there is an increase in the total utility of the system."

Compared to the single-robot system, the cooperative multi-robot system is distinguished by the following aspects:

- Multi-robot systems can accomplish some inherently complex tasks that cannot be accomplished by single-robot systems, e.g., robot soccer.
- Multi-robot systems can enhance performance by working cooperatively.
- Multi-robot systems are more robust than single-robot systems.
- As compared with an expensive and multi-functional robot, the cost of a team of simple and heterogeneous robots may be cheaper.

Generally, multi-robot systems can be categorized by the control architecture (centralized or distributed), robot differentiation (homogeneous or heterogeneous), and cooperation level (low to high). In the following, each one of these classifications is explained in detail.

2.2.1.1 Centralized and Decentralized

With respect to the control architecture or system infrastructure, systems can be classified as centralized or decentralized. In centralized control, a commander (a leader robot or a host PC) gives commands to every robot; whereas in decentralized control each robot makes decisions by itself and works with other robots without the need for such a commander. In most research, decentralized (distributed) control is preferred because it can achieve better robustness.

2.2.1.2 Homogenous and Heterogeneous

Depending on the differences in the types of robots that are being deployed, systems can be classified as homogenous and heterogeneous. If all robots are identical, the team is homogeneous; otherwise, it is heterogeneous. In a heterogeneous robot group, individual robots usually have different physical configurations and capabilities.

2.2.1.3 Action-Level Cooperative or Task-Level Cooperative

Regarding the level of cooperation, multi-robot systems can be classified as task-level cooperative and action-level cooperative (Tangamchit et al., 2002). In task-level cooperation, the mission is broken down into simpler tasks; each robot chooses different tasks (roles) and behaves differently, according to the task allocated to it. Action-level cooperation does not differentiate between the behaviors of the robots. It is usually accomplished by enabling the robots to work in parallel. The term “cooperation” in action-level cooperation only refers to the fact that a robot will not impede others, e.g. collision. For instance, in a robot soccer team, task-level cooperation

allows the robots to take on different tasks (behaviors), such as defending, passing, and shooting. On the other hand, if the cooperation is at the action level, the mission is not divided and all robots try to achieve the same goal: get the ball and then kick it towards the goal. Task-level cooperation typically has superior performance as compared to action-level cooperation.

2.2.2 Research Issues

For multi-robot systems, the decentralized control methodology is more applicable and reliable. This is due to the following reasons:

- For a large robot team, it is difficult and sometimes impossible to gather the information from all the robots to the central commander and then send commands to the robots. Even if such information aggregation and dissemination are possible, the transmission delay is large (and unreliable in a wireless environment) and thus can hardly be applied to time-critical tasks.
- Compared to the centralized control architecture, the decentralized control architecture is more robust and reliable. Decentralized multi-robot systems can continue working even when one or several robots are out of order.

Due to the above reasons, the primary focus of this research on cooperative multi-robot systems is on a decentralized (distributed) control methodology that can enable the robots (either homogeneous or heterogeneous) to cooperate at the task level. To achieve this, research issues such as mission decomposition, task allocation, robot coordination and conflict avoidance have to be studied. In addition, since the desired control methodology is distributed in nature, it is important to enable and utilize appropriate intercommunications among the robots.

Since the aim of this study is to design practical multi-robot surveillance systems, it is important and necessary to have *a priori* knowledge of the related work on the above-mentioned critical research issues. In the following, the typical control methodologies and communication techniques for multi-robot systems are introduced.

2.2.3 Control Methodology

In robotics research, four basic policies are usually used to control the robot: reactive, deliberative, hybrid, and behavior-based control. They are summarized by Mataric (2001) as follows:

- Reactive control: do not think much, just act depending on sensor inputs.
- Deliberative control: think first, and then act.
- Hybrid control: think and act independently at the same time.
- Behavior-based control: think of the role to act.

The characteristics of the mission and the environment, and the capabilities of robots determine the suitable control policy. Reactive, deliberative, and hybrid controls are usually used in single-robot systems and are able to achieve action-level cooperation in multi-robot systems. Behavior-based control is more complex, but it can achieve task-level cooperation in multi-robot systems. Therefore, behavior-based control is the most prominent control methodology that is applied to cooperative multi-robot systems, and is the focus of this thesis.

2.2.3.1 Reactive Control

A reactive control policy enables the robot to make decisions by reacting to the sensor inputs using a set of simple rules (such as artificial neural networks or fuzzy logic). As only local sensor

input is usually considered, reactive control has a small response delay and is suitable for time-critical or large-scale applications. The drawback of reactive control is the local minima problem, which may affect the global optimization. This is mostly due to the partial observation of the local sensing.

In distributed robotic systems, reactive control is applied in a way such that each robot makes its decision based on the local discovery of the states. There is very little, if any, information-sharing between the robots and they do not make decisions together. A typical reactive control scheme is Artificial Potential Field (APF)-based reactive path/motion planning (Khatib & Le Maitre, 1978). In this algorithm, undesired objects (obstacles or other robots) are mapped as repulsive force sources, and desired objects (target or destination zone) are mapped as attractive force sources; the robot can then move under the summation of these forces to approach the destination without collisions. However, as only local sensing is available, APF-based control is prone to the local minima problem, in which the summation force is zero and the robot can no longer continue to move towards the destination.

2.2.3.2 Deliberative Control

Deliberative control is an off-line control policy. It decides a sequence of actions for the mission or task by considering all possible sensory inputs and environmental changes before the robot starts to move. A typical deliberative control policy is the off-line path planning problem. Before the robot begins its motion, the best path from the starting point to the destination is determined based on the map. The robot can then move according to the designated path without any changes in its path during the execution.

As the decisions are made with the knowledge of all possible states of the environment, global optimization can be achieved. However, the drawback deliberative control has is its heavy computation: the system designer must anticipate all possible environmental encounters and program the corresponding actions. This is not practical for unknown or partially known environments. In particular, an action sequence is difficult to establish when the system is very complex. Furthermore, because the action sequence is determined before the robot commences its movement, the robot is unable to handle dynamic environments.

2.2.3.3 Hybrid Control

As reactive control is simple but non-optimal (due to local minima), and deliberative control is optimal but inflexible, it is intuitive to combine both of them to generate a hybrid control methodology which is both simple and optimal. Currently, in single-robot system research, the hybrid control policy of reactive and deliberative control is dominant because it can utilize deliberative global decisions to achieve global optimization, and use reactive responses to handle the changes in the environment. An example of hybrid control is a path-planning approach (Low, 2002; Low et al., 2003). The hybrid control algorithm first designs an optimal path (e.g., go from room 1 to room 3, and then room 6) for the robot. The robot then moves along the designated path, and at the same time, the robot also monitors and reacts to the changes in the environment using a neural network controller and local sensing.

The main challenge of hybrid control is to arbitrate and coordinate the reactive and deliberative control components (commands), which can be overcome by putting these components into different layers. The critical components, e.g., real-time reactive obstacle avoidance, are in the lowest layer (or highest priority), while the functional or uncritical components are in the higher layers (lower priorities). In such a control architecture, low level means high sampling rates and

high priorities, e.g., joint servo rates. High-level algorithms are more complex and hence have lower sampling rates, e.g., planning, vision, etc.

2.2.3.4 Behavior-based Control

In multi-robot systems, it is highly desirable that each robot has its individual appropriate response (behavior) to accomplish the mission, instead of acting in the same manner as other robots. This is the basic concept of behavior-based control. Behavior-based control typically tries to break down the mission into tasks (behaviors), and each robot then chooses the best or most suitable task (behavior) to act upon. Therefore, a behavior-based control methodology enables the robot team to achieve high levels of cooperation. As behavior-based control "scales well to collections of robots, resulting in robust, adaptive group behavior" (Mataric, 2001), it is the most extensively studied control policy for cooperative multi-robot systems.

As compared to reactive, deliberative or hybrid control methodologies, behavior-based control is a high-level control method that makes high-level decisions based on abstractions of low-level states and actions. While other control methodologies are usually hardware-centric, in that the controller tries to find the optimal link between raw inputs and outputs, behavior-based control method is nearly hardware independent because it focuses on the control strategies to determine the best high-level actions for the high-level states (as depicted in Figure 2-2).

The advantages of behavior-based control are listed as follows:

- Behavior-based control is hardware-independent. Whenever the hardware changes, the behaviors do not need to be changed. Instead, the system designer only needs to modify

the "abstraction" or "concretization" process to update the definition of high-level states or actions.

- Behavior-based control is easily understood by the system designer because it is written in high-level human linguistics which show clear rationales, e.g., if "find targets" (high-level state), then "track it" (high-level action). On the contrary, it is hard to display or summarize the rationales in low-level control methodologies, e.g., a neural network controller.

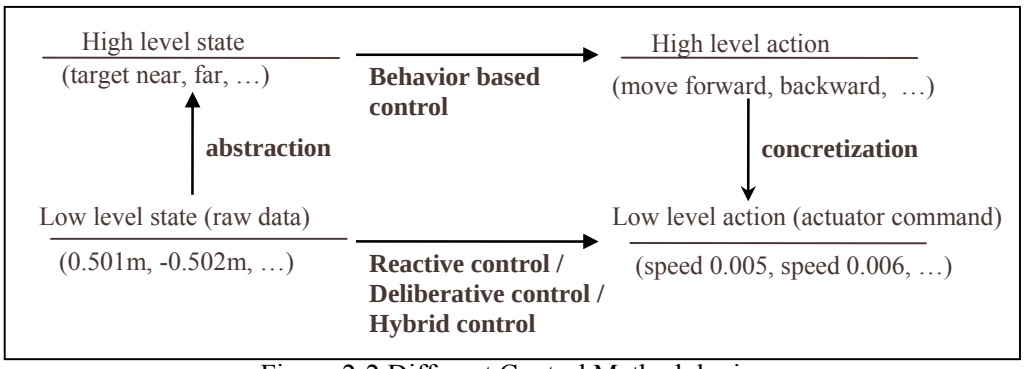

Figure 2-2 Different Control Methodologies

In behavior-based control, the essential problem is to define the high-level states and actions, and then assign behaviors (link from state to action) to robots, with the consideration of resource usage, environment states, robot capability and fault tolerance. The implementation of behavior-based control ranges from simple finite state automata (Kube & Zhang, 1994; 1996) and motivation-based behavior selection (Parker, 1996; 1998; Murphy et al., 2002; Werger & Mataric, 2000) to market-based contract methods (Botelho & Alami, 1999; Gerkey & Mataric, 2002; Zlot & Stentz, 2006).

Finite state automata (FSA) is the simplest behavior-based control methodology. It is based on the state transition diagram where each robot detects the state and executes the corresponding action for that particular state. Kube & Zhang (1994, 1996) proposed FSA control for a group of

box-pushing robots. The robots cooperate using local sensing without the need for intercommunications. However, as the robots rely solely on local information, global performance is limited and the robots are sometimes trapped in deadlock. In general, the advantage of FSA lies in its simplicity and scalability. The disadvantages are that FSA is suitable for homogeneous robot teams and is difficult to achieve high-level cooperation. This is because all the robots behave identically under FSA control: if they are in the same state, they will execute the same actions according to the state transition diagram.

To enable a heterogeneous robot team to accomplish complex missions, motivation-based control has been proposed and studied to address the limitations of FSA. In motivation-based control, the robots are assigned different preference levels (or "motivations") for each task. Via interactions among robots, each task is to be assigned to the robot with the strongest preference or motivation.

Among the motivation-based control methodologies, ALLIANCE (Parker 1996, 1998) is arguably the first and most well-known methodology. In ALLIANCE, to achieve cooperation in task selection, each robot has two motivations, impatience and acquiescence. When a robot is executing a particular task, it will become increasingly impatient to continue on with the task. If the impatience increases beyond a threshold, the robot will give up the task. By acquiescence, the robot will have less inclination to execute a task if this task is being executed by other robots. If the acquiescence level is higher than a threshold, the robot will not execute the task. The increase/decrease in impatience and acquiescence of each robot is dependent on its individual ability; therefore heterogeneous robots are able to achieve specialization. Besides ALLIANCE, other approaches have been developed to improve the effectiveness of task allocation by adding more factors in addition to "impatience" and "acquiescence". Murphy et al. (2002) introduced "emotion"-based control, whereby each robot performing a task has one of the following four emotions – "happy", "confident", "concerned" or "frustrated". Using intercommunications, the

emotion of each robot is broadcast to the whole team; therefore the robots can select the most suitable behavior with the consideration of other robots. Instead of using "emotions", Werger & Mataric (2000) proposed the Broadcast Local Eligibility (BLE) method for task allocation. In this approach, "robot utility" is given by the subtraction of the cost in performing a particular task from the quality achieved by the robot in executing that task. High "robot utility" means the robot is good at the task. Each robot is required to send its robot utility to other robots, and thus they can select the appropriate task to do according to the ranking of robot utility.

In general, motivation-based task allocation takes into account the capabilities of all robots; therefore it is suitable for both homogeneous and heterogeneous robot groups and it is also able to achieve better task allocation than finite state automata. However, motivation-based task allocation usually requires the sharing of "motivational values" of robots; therefore either explicit intercommunications (emotion-based approaches, BLE methods) or the ability to observe other robots and task progress (ALLIANCE approach) is required.

While the motivation-based control methodology focuses on local motivation, the market-based contract method considers the negotiations among robots. The market-based method stems from the economical solutions for allocating limited resources: the resource will be granted to the bidder who offers the highest price (Botelho & Alami, 1999; Gerkey & Mataric, 2002; Zlot & Stentz, 2006). Like the motivation-based control methodology, the market-based contract method can be applied to both homogeneous and heterogeneous robot groups.

One representative market-based control methodology was proposed by Gerkey & Mataric (2002). In the robot group, there exists an auctioneer who is in charge of broadcasting task requirements and specifications whenever a new task is discovered. Upon receiving this advertisement, each of the robots will send a bid to the auctioneer regarding this task. The winner will be offered a time-

bounded contract from the auctioneer. The time-bounded contract means that the auctioneer will constantly monitor the progress of the task; if the task is not accomplished after a certain time period, the auctioneer will cancel the contract with this robot and then host a new auction to assign the uncompleted task to another robot.

The market-based control methodology is flexible and effective for multi-robot task allocation. However, an arbitrator is usually needed to send out task requirements and assign tasks based on the response from the robots. This arbitrator may be an outsider robot or computer (Gerkey & Mataric, 2002), or a common robot that is part of the team (Zlot & Stentz, 2006). This limits the robustness of the system because the robot team cannot continue working if the auctioneer functions incorrectly. In addition, the posting of task (from auctioneer), sending of bid (from robot), and signing of contract (between auctioneer and the selected robot) require explicit and rapid information exchange among robots, which leads to the dependency on high-performance and expensive communication techniques.

In summary, the behavior-based control methodology is a powerful strategy for controlling multi-robot systems. It can achieve high-level cooperation by taking the environment state, robot ability and task property into consideration. However, some difficulties impeding the application of behavior-based control are as follows:

- Behavior-based control usually requires deliberative design work, e.g., the design of a state transition diagram (FSA control), emotions (motivation based behavior selection), or negotiation rules (market-based contract method). Such design work is especially difficult when the mission is complex and the robot team is heterogeneous.
- Most behavior-based control methodologies require explicit and rapid intercommunications among robots. However, in real-world applications, it is difficult to

achieve fast and error-free communications, especially for multi-robot systems in large environments.

This study aims to overcome the above mentioned limitations of behavior-based control. It is important to develop distributed, simple, effective, and robust control algorithms for cooperative surveillance robots while leveraging the advantages of behavior-based control at the same time. The control algorithm should also be able to minimize the need for explicit and intense intercommunications.

In some extremely difficult cases, e.g., tracking unknown targets in unknown environments, it is highly desirable that some types of machine learning algorithms can be applied such that the robots have the intelligence to learn how to cooperate without the need for deliberative human design.

2.2.4 Communications

In cooperative multi-robot systems research, communications is one of the key issues because it is the basis of cooperation. Through communications, the robots are able to share three types of critical information (Cai et al., 1997):

- Task description: A robot needs to negotiate with others to determine its task and corresponding actions.
- Robot states: In some applications, the states of other robots are critical information in making the decision, e.g. in a robot soccer team, a passer needs to know the location, velocity and heading of the shooter. However, such information is usually not obtainable by local sensing.

- Environment states: With communications, a robot can sense the environment more completely, accurately and quickly by sharing the sensor data of other robots.

In robotics, "communications" is not solely limited to wired or wireless communications. It can be categorized into two types (Kurabayashi, 1999): indirect and direct communications. Indirect communications is termed "implicit" communication, while direct communications is termed "explicit" communication. Indirect communications is usually realized by sensing, e.g. detecting the actions of other robots. With the exception of some research developments such as the "message board" (Fujii et al., 1996), indirect communications does not require data transmission among robots; whereas direct communications transmits data between robots using wireless transceivers, infra-red sensors, acoustic modems, or any other means.

Indirect communications is often used for simple robots or large robot teams. For indirect communications, the robots share information by the sensing of the environment. In this respect, the term "communication" is similar to the term "interaction". If the sensing of a robot is limited (either in quality or functionality), the performance of such communications (and cooperation) might be poor. For instance, in a box-pushing robot team with only indirect communications (Kube & Zhang, 1996), a robot has to make decisions locally. However, as the robot is unaware of the states of other robots, it cannot avoid contentions with other robots, which can lead to undesired stagnancy.

Direct communications allows the robots to transmit data among themselves. Therefore, the robots can share information more completely, accurately and quickly. Direct communications is usually implemented using Radio Frequency (RF)-based wireless connections (Cai et al., 1997; Hoshino et al., 2000; Ichikawa et al., 1993; Ikenoue et al., 2002; Ohkawa et al., 1998). Other

technologies such as acoustic and infra-red connections can also be used for direct communications (Arai et al., 1997).

In general, indirect communication can be easily implemented; however, it only exchanges implicit information. Therefore, the robots are difficult to achieve high-level cooperation. On the contrary, direct communications is explicit, accurate and fast. However, it requires expensive communication device or equipment. In addition, it is infeasible to enable satisfactory communications for large robot teams or in large areas due to contention and interference.

The aim of this study is to design efficient and practical surveillance systems. To achieve satisfactory performance, the cooperative robots should have some form of direct communications to realize high-level cooperation. On the other hand, the direct communications should be optimized and cost of the system minimized.

2.2.5 Summary

In cooperative multi-robot systems research, the main objective is to design decentralized (distributed) control algorithms that can enable a group of robots (homogeneous or heterogeneous) to cooperate at the task-level to accomplish common missions. The main research issues include mission decomposition, task allocation, robot coordination and conflict avoidance. In addition, the intercommunication techniques among robots are critical for distributed control algorithms.

In this section, the representative control methodologies for cooperative multi-robot systems have been reviewed. Among these methods, the behavior-based control methodology is suitable and powerful, but it requires excessive human deliberative work and is usually mission-specific (e.g., it cannot be easily reused if the environment or the robot changes). This motivates the study of

developing simple, practical and powerful surveillance algorithms for cooperative robots. Furthermore, in extremely complex scenarios, e.g., tracking unknown targets in unknown environments, it is highly desirable that some machine learning algorithms be applied to enable robots to learn cooperation without the need for deliberative human design.

In this section, the intercommunications issue for cooperative multi-robot systems is also introduced. This is non-trivial because the distributed robots need to share information on the task, the resource, and the environment in order to achieve cooperation. It is highly desirable that the proposed communication techniques facilitate efficient cooperation with minimal cost.

3 PROPOSED SURVEILLANCE SCENARIO AND ROBOT SYSTEMS

3.1 Application Scenario and Environment

A typical application scenario that involves surveillance in unknown environments is the search and rescue operation for victims inside a building after an accident, e.g., terrorist attack. For such an application environment (building), there are usually some difficulties which hinder surveillance, such as the following:

- Environment is unknown. Map information is not available, or the environment changes a lot due to the damage caused by the accident.
- Location information is not available. Consequently, it is hard to localize robots or sensors inside the environment (building).
- The conditions are harsh, with excessive heat or pressure, and the possibility of chemical explosions, etc.

In this study, an autonomous multi-robot system is developed to perform the surveillance tasks without humans having to take the risk of entering such an environment. In this surveillance system, some wireless beacons are placed at designated corners outside the building. They function as gateways though which people communicate with the static sensors and robots that have been dispatched into the building. In addition, the beacons can be used for the localization of sensors, robots and targets. The static sensors and mobile robots may be deployed (or thrown) in the building through windows and/or openings; therefore they are likely to be clustered in some areas of the environment. Obviously, such initial placement may not be optimal because the sensors and robots should be dispersed in the entire environment. To achieve satisfactory surveillance, the autonomous robots need to move around adaptively to change the topology of

the network for sensing, and perform the search and rescue tasks at the same time. On the other hand, the static sensors can detect the physical conditions of the environment (building), report the sensed information back to the beacons (or controller), deliver messages for the robots, or even cooperate with the mobile robots to perform the surveillance tasks.

With respect to this application scenario, the work reported in this thesis aims to find a series of practical multi-robot surveillance solutions that can be applied to the real world. Therefore, the application environment should meet the following requirements:

- The environment should be realistic and generic. In this thesis, we consider a bounded 2-dimensional (2D) indoor environment, which can also be extended to 3D or outdoor environments.
- The environment should be large. It is not meaningful to apply multiple robots in a small environment where the surveillance task can be easily achieved by a single robot.
- The internal conditions, e.g., walls, obstacles, of the environment should be unknown. This makes the surveillance system meaningful to many critical applications such as search and rescue, disaster response, or civil defense.

3.2 Surveillance System

In practice, the surveillance of a large environment is fraught with the following difficulties:

- Complex and powerful mobile robots can provide the required functionality. However, it is not realistic to deploy many sophisticated robots to monitor the entire region due to cost constraints.

- Simple and cheap static sensors can be deployed in large numbers to occupy the entire environment. However, they do not have sufficient capabilities for high-level environment monitoring, e.g., video capturing of intruders.
- The short communication range (with respect to the environment size) limits the information sharing among robots and sensors; therefore they may not be able to transmit information on the environment to beacons (or controller) that are placed outside.

To address the above mentioned problems, this study proposes a hybrid surveillance system with infrastructure-less communication methodologies. The system has the following properties:

- The system is hybrid. It consists of a small number of mobile robots and a large number of static sensors[2].
- The robots and sensors are able to inter-communicate. They are equipped with wireless transceivers and apply ad hoc networking technology (Frodigh et al., 2000) to send and receive information.

3.2.1 Simulation System

The simulated application environment and surveillance systems are shown in Figure 3-1. In the large unknown environment, four beacon nodes (sinks) are placed at the four corners, and many static sensors are randomly deployed inside the environment. The static sensors perform low-level monitoring tasks such as sensing of temperature, sound, light condition, or other information. These sensors use self-organizing ad hoc communication techniques to transmit the sensor data to the monitoring center through beacon nodes (sinks). Since the deployment of static sensors is

[2] For the purpose clarity, in the following parts of this thesis, "robot" is synonymous with "mobile robot", and "sensor" is synonymous with "node", "static node" or "static sensor".

random and the environment is unknown, it is not practical to assume that the locations of the sensors are known. In this case, the four beacon nodes (sinks) can help the static sensors to locate themselves after the deployment phase by using the hop-count-based localization method, which will be further discussed in Chapter 6.

In addition to the static sensors, some mobile robots work in the environment to perform some high-level surveillance tasks, such as exploration and searching for targets, tracking of targets, etc. The robots are aided by the static sensors, which help the robots to deliver information (by ad hoc communications), and also provide useful information (for searching targets). On the other hand, the mobile robots also improve the static sensors' data collection by enhancing the communications of the ad hoc network, and improving the accuracy of the hop-count-based localization.

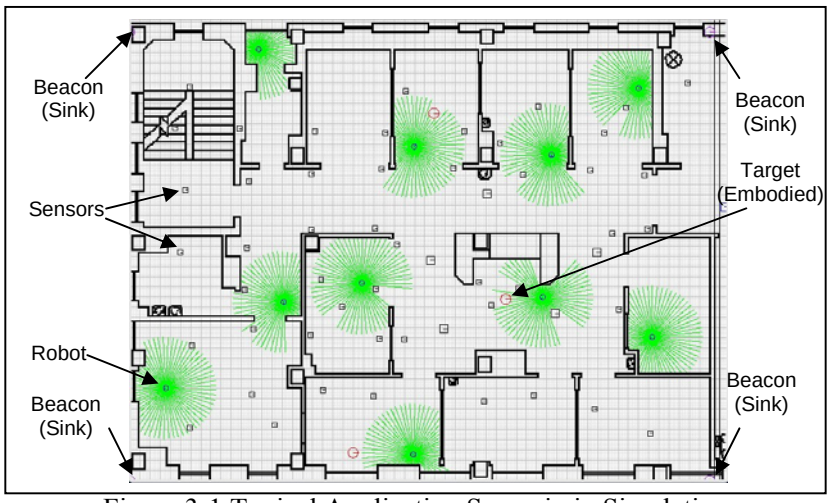

Figure 3-1 Typical Application Scenario in Simulation

Because the focus of this thesis is the control problem of cooperative robots, it is important to make realistic and reasonable assumptions about the robots. In this study, one assumption is that the mobile robots have the following features:

- The robot has limited panoramic sensor range (as indicated by the short lines starting from robot in Figure 3-1).
- If an object is within the sensor range of the robot, it will be able to detect the distance/orientation to this object. In addition, the robot can differentiate and identify this object as a target, obstacle or other robot. For example, robot *A* can detect that target *B* is at true bearing 25 degree and is 3 meters away.
- The robots and sensors have wireless transceivers, which they use to send/receive information to/from others. The communication is performed using ad hoc networking techniques.
- The targets can either be embodied (a real object to be monitored) or virtual (a critical region for observation or other purposes). In addition, the targets may be static or mobile. If the target is static, the robot does not need to track it after it has been found; if the target is mobile, the robot has to track the target continuously to keep it under observation.

Another assumption is that the robot system does not have the following capabilities or functionalities:

- The ability to build a map of the environment (due to limited sensing ability, computation power, memory storage, and communication speed). Comparing to single-robot systems, it is much more difficult to build a map by multi-robot systems. To match and fuse the data obtained from different (especially heterogeneous) robots, the system should have high computation power or memory storage. This is challenging for small and simple robots. If centralized processing power (e.g., control PC) is used, the transmission of data will be a problem. In large-area environments, a group of robots may not have the high speed and bandwidth communication channels to the control center.

- The ability to localize by special sensors, e.g., GPS. In this application, the robots obtain location information using a hop-count-based localization technique, which largely depends on the distribution of the sensor nodes in the environment.

3.2.2 Experiment System

In this study, the proposed algorithms have been implemented in two real-world systems. One experiment system is done with compact-size robots (as small as a cup) in small environment (single room size); then the experiment is extended to middle-size robots in large environment (multiple room size).

The small experimental environment is shown in Figure 3-2. This is a 6m x 5m area with some internal walls. The small robots are shown in Figure 3-3. This is MRKIT robot made by Alpha Innovation Inc[3]. The robot's processor is a Hitachi H8/3064F 16bit Micro-controller running at 20MHz. It has internal flash memory of 256 KB and external memory of 512 KB. The MRKIT robot is equipped with 12 short-range (less than 40cm) Infra-Red (IR) sensors for obstacle detection. It also has a digital compass that can detect the heading. There are two serial ports on the robot; one of them is used to connect to the wireless transceiver. The robot has two stepper motors for each wheel and its maximal speed is about 0.8m/s in forward motion.

[3] Alpha Innovation Pte. Ltd. is a developer and supplier of educational technology and robotic learning systems in Singapore. (http://www.ai.com.sg/)

Figure 3-2 Small Experimental Environment

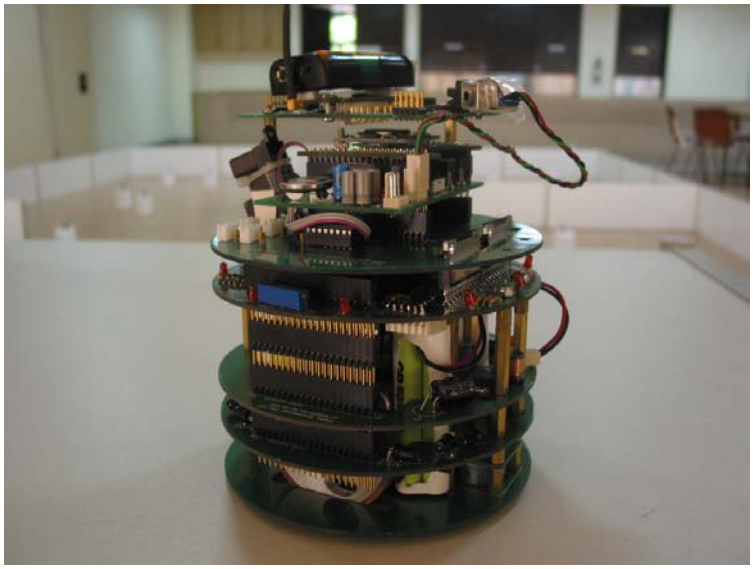

Figure 3-3 MRKIT Robot

The extended experimental environment is shown in Figure 3-4. This is a 15m x 10m large area with some internal objects. The middle-size robots are shown in Figure 3-5. This is the Koala robot made by K-Team[4]. The robot has a 400MHz PXA255 processor. It has internal flash

[4] K-Team Corporation is a Swiss company that develops, manufactures and markets mobile robots for use in advanced education and research. (http://www.k-team.com/)

memory of 32 MB and external memory of 64 MB. The Koala robot is equipped with 16 Infra-Red (IR) sensors (maximum range is about 120cm) for obstacle detection. It also has a digital compass that can detect the heading. There are three serial ports on the robot; one of them is used to connect to the wireless transceiver. The robot has six wheels driven by two DC motors. The maximum forward-motion speed is about 2m/s.

Figure 3-4 Extended Experimental Environment

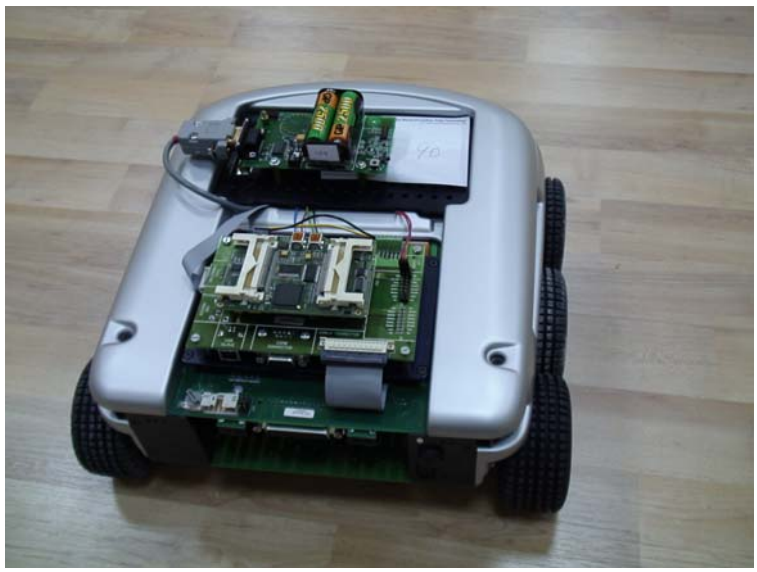

Figure 3-5 Koala Robot

For experiments, the wireless transceiver is MICAz mote made by Crossbow Technology[5] as shown in Figure 3-6. It is a 2.4 GHz, IEEE 802.15.4-compliant module designed for low-power wireless sensor network applications. It runs under TinyOS (Levis et al., 2004) and has a data rate of 250 kbps. The mote has a transmission range of 70~100m in an outdoor and 20~30m in an indoor environment. In this study, the communication range of the wireless transceiver is limited to 1.5 to 2 meters for the small experiment environment and 2 to 3 meters for the extended experiment environment. The mote can work alone, or be connected to a computer, or be connected to robots (MRKIT or Koala) via RS-232 serial interface.

Figure 3-6 Wireless Transceiver

The details of the simulation and experimental platforms and settings are introduced in the following chapters.

[5] Crossbow Technology Inc. is a supplier of sensor systems for aviation, land, and marine applications. It provides the wireless sensors. (http://www.xbow.com/)

3.3 Ad Hoc Communications

Communication is one of the most important research issues in surveillance: the observer (robot/sensor) has to report information about the environment and targets to the monitoring center in a timely and reliable manner. If the information (data) is unable to reach the monitor center, the surveillance is unsuccessful because people cannot obtain the knowledge of the environment. To achieve the desired high level of cooperation among robots, or between robots and sensors, communication is also necessary. Much related robotics research (Emery et al., 2002) uses long-range wireless communication techniques to transfer messages between robots/sensors and the monitoring center. However, such work is not very applicable in many real-world applications for the following reasons:

- In large-area surveillance, the size of the environment (on the order of tens of km) may be too large for wireless transmission (on the order of km). On the other hand, for small-area surveillance, especially in indoor environments, the obstacles (e.g., walls) may attenuate the radio waves and shorten the transmission range such that direct communication cannot work properly. For example, normal Radio Frequency (RF) transceivers can only transmit signal among nearby rooms in a building. However, to accomplish the surveillance task in the entire building, such transmission range is not acceptable.
- Long-distance direct communication may consume excessive power. This is undesirable for small robots and sensors, which have energy constraints. Also, for long-time continuous surveillance, the communication power should be limited to extend the battery life.
- When many robots/sensors are monitoring the environment or targets simultaneously, they may all transfer data at the same time. This can possibly incur communications

interference and thus lower the throughput of the communication network. In the worst case, all robots and sensors may transmit at the same time and the monitoring center may only hear the corrupted communication messages due to interferences.

- It is not always necessary to establish communications between robots that are very far apart. For example, when searching for a target in a sub-region of the environment, only the robots inside the sub-region should be involved. Therefore, it is reasonable to limit the communication to this sub-region.

To address the above mentioned problems, ad hoc communication techniques are applied in the proposed practical surveillance application. Compared to traditional structured networks, ad hoc networks (Frodigh et al., 2000) are well known for their self-organizing capabilities, such as the following:

- An ad hoc communication network is a "large network with small nodes". The communication range of each node is short; a node can communicate with a distant node using intermediate nodes to relay the communication message from the source to the destination. Multi-hop communication is typically used between any two non-adjacent nodes.
- The ad hoc communication network is infrastructure-less, so that all the communication nodes have equal capabilities. There is no central commander to gather messages or control the behavior of other nodes.
- The ad hoc communication network is self-organizing, such that the nodes try to find the shortest or most suitable route by proactive or reactive routing algorithms.
- The ad hoc communication network is distributed (decentralized) and cooperative. Each node can be a source, destination, or an intermediate node which forwards packets for other nodes in the network. They make routing decisions independently of each other.

The ad hoc communication architecture is depicted in Figure 3-7. In this figure, a static sensor (source) intends to send information to one of the sinks (destination). It does not directly communicate to the monitoring center, as it is far away; instead, it passes the information to its neighbor (intermediate node), which forwards this information to its neighbor (another intermediate node). This forwarding process will continue until the information reaches the sink (destination). To achieve such self-organizing communication, a routing protocol such as the Ad hoc On-demand Distance Vector (AODV) routing algorithm (Perkins & Royer, 1999) is required. However, as the focus of this study is on cooperative robotics, details of the communication techniques are beyond the scope of this thesis.

In ad hoc communications, it is important to enable and maintain the connectivity of the network of sensors. Two sensors are considered to be connected if they are within communication range of one another, or have intermediate sensors within their communication range to help forward information. If two sensors are physically far apart from each other, and they do not have any intermediate sensors to ferry information, they cannot communicate. In this case, there is a "communication gap" between the two sensors. As shown in the left figure of Figure 3-8, sensor *C* is connected to sensor *A* even though they are not within the communication range of each other, because they can communicate through sensor *B*. In the right figure, sensor *C* is not connected to sensor *A*; in this case, there exists a "communication gap" between them. If another sensor is deployed in this communication gap, it can enable connectivity between sensors *A* and *C*.

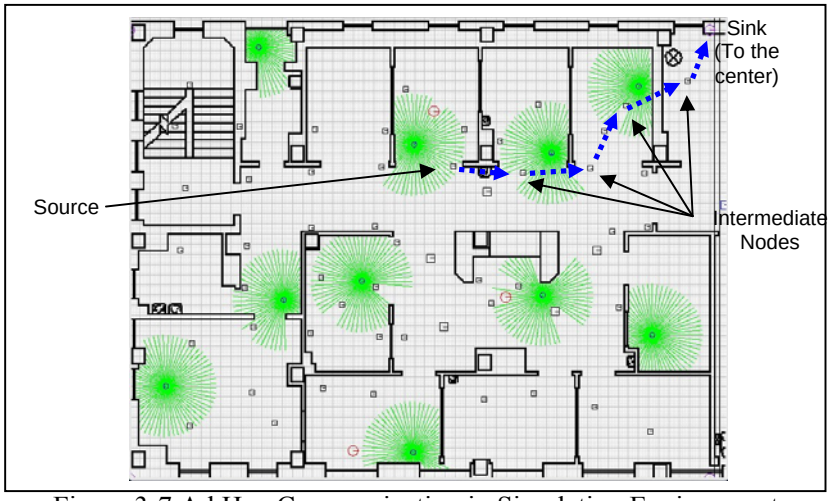

Figure 3-7 Ad Hoc Communication in Simulation Environment
(The source sensor node sends a message and the intermediate nodes help to forward this message to the sink)

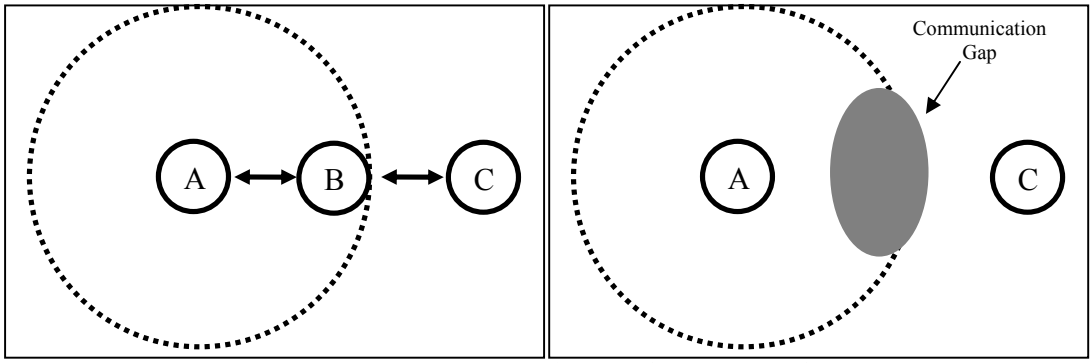

Figure 3-8 Connectivity in Ad Hoc Network.
In the left figure, sensor *A* can "talk" with sensor *C* through sensor *B*; in the right figure, sensor *A* cannot "talk" with sensor *C*. (Dot Circle: Communication Range of Sensor *A*; Gray Ellipse: Communication Gap)

3.4 Targets in the Environment

In the application environment, there are two kinds of targets – embodied targets and non-embodied (virtual) targets. The embodied targets are the objects to be observed or monitored, e.g., victims, intruders, fire sources, etc. They can be either mobile or static. The non-embodied (virtual) targets are the communication gaps in ad hoc communication networks. Communication gaps can badly affect the performance of the network and therefore they are also the targets for the robots to detect. If the robots find such communication gaps, they may stop at the gap to relay

messages or deploy new static sensors (e.g., by dropping them) at the gap to improve the network connectivity.

In the following parts of this thesis, the term "targets" refers to both the embodied objects and virtual communication gaps. Both of them should be found and handled by the cooperative robots.

3.4.1 Embodied Targets

In the surveillance system, the robots can detect the embodied targets by using the typical sensors. For example, a robot can find a target using a laser scanner, video camera or heat sensor. A generic system is considered whereby the sensor to detect embodied targets is not specified. It is assumed that when a target is within a certain distance and is within the "line-of-sight" of the robot, the robot can detect and identify the target.

3.4.2 Virtual Targets

As compared to traditional surveillance systems, the virtual targets are specific in our study. We consider the communication gaps to be virtual targets for robots to search due to the following factors:

- In the surveillance of large environments, the ad hoc sensor network is practical and feasible. It is non-trivial to maintain and enhance the ad hoc network connectivity because the connectivity can affect the quality and performance of information delivery. In some cases, it is even more important to maintain the network connectivity (find communication gaps) than to find the embodied targets.

- Hop-count-based localization is applied in this surveillance system. To obtain satisfactory localization accuracy, it is important to increase the network connectivity, e.g., fill the communication gaps or increase the number of nodes at particular regions.

As compared to embodied targets, the detection of virtual targets is different because it cannot be visually seen and is related to the network connectivity. In the proposed surveillance scenario, the robots can detect the communication gaps by comparing the hop-count values (to the sinks/beacons) of their neighboring static sensor nodes.

As introduced in Section 3.2, there are four reference nodes (beacons) deployed at the corners of the surveillance environment. During the setup/initialization phase, each beacon broadcasts a localization message to the network inside the environment. When a sensor node receives this message, it will note the hop-count value (i.e., number of intermediate nodes that are used to forward the message to the sink) and rebroadcast the message after incrementing the value by one (hop). Every sensor keeps a record of its hop-counts from all the beacons (initial values are set to a large number, e.g. 255; consequently a sensor that has no route to a particular beacon will have 255 as the corresponding hop-count.)

Based on the above mentioned hop-count propagation mechanism, a pair of nodes in the deployed sensor network that are within range of each other will have their 4 hop-counts differing by not more than one. As a robot moves, it constantly sends requests to the neighboring sensor nodes for their hop-count information. When it receives large differing 4-tuple hop-counts, it identifies the area as a communication gap. For example, assume there are two static sensors *A* and *B*. Sensor *A*'s hop-count is <3, 255, 6, 255>, which means *A* is 3 and 6 hops away from reference nodes 1 and 3, but not connected to reference nodes 2 and 4. Sensor *B*'s hop-count is <255, 7, 255, 9>, which means *B* is 7 and 9 hops away from reference nodes 2 and 4, but not connected to reference

nodes 1 and 3. These two sensors are disconnected and cannot communicate, even through intermediate sensors. Both sensors *A* and *B* will periodically broadcast their hop-counts message to the neighborhood. If a robot receives both messages from *A* and *B*, it can compare the hop-counts, and identify that it is at a communication gap between disconnected sensors *A* and *B*. In this case, the robot can deploy a new static sensor at that location to "bridge" sensors *A* and *B* to improve the network connectivity.

3.5 Overview of Surveillance Tasks

To summarize, the cooperative robots in the proposed surveillance system have the following surveillance tasks:

- From a randomly selected part of the unknown environment, explore the environment to search for static targets (stationary objects or communication gap regions) or mobile targets (mobile objects).
- When a static target is found, the robot will maintain the target within a predefined range. The robot will then localize the target and transmit this information to the base station (for embodied targets) or deploy new sensors at virtual communication gaps (virtual targets) to enhance network connectivity.
- When a mobile target is found, the robot will track it for continuous observation.
- When required, the robot will deploy new sensors, or stay at sparse network areas to improve the hop-count-based localizations.

In the simulation platform, the robots execute all the above-mentioned tasks. For the experiment platform, the robots only perform the tasks of exploration and target searching, as our current robots are not equipped to distinguish and identify mobile targets, e.g., using cameras to track

targets. However, since the proposed algorithm is realistic, it could be justified that the real robots can achieve their tasks with performance similar to that demonstrated in simulation.

4 MULTI-ROBOT EXPLORATION AND TARGET SEARCHING

In multi-robot surveillance, the first step is to find the target(s) to be surveyed and monitored. This involves the study of exploration and target searching. Both "exploration" and "target searching" can help to find desired targets in unknown environments. However, there are some key differences between "exploration" and "target searching". Exploration is coverage-centric and aims to have a complete view of the environment using the lowest cost in time or traveling distance. If the robots can explore more regions of the environment, they are more likely to find the targets. In particular, all targets can be found if they are static and the robots are able to explore the entire area. On the other hand, target searching is target-centric and aims to find the targets using the lowest cost in time or traveling distance. A good searching algorithm may not necessarily try to maximize the coverage of the environment in order to find the targets. Instead, the robots are encouraged to utilize useful information such as target visiting history to search for targets in promising areas where targets are likely to appear.

While exploration and target searching have different research purposes and may cause the robots to behave differently, both of them can be applied to surveillance tasks to help find the targets to be observed. In this study, both exploration and searching algorithms are developed and implemented (Seah et al., 2005, 2006). In the following parts of this chapter, the exploration algorithms and target-searching algorithms are introduced in Sections 4.1 and 4.2, respectively. Section 4.3 discusses the implementation of these proposed algorithms in simulations and hardware tests, and presents the corresponding results and discussions. Finally, Section 4.5 concludes this chapter.

4.1 Exploration

As introduced in Section 2.1.1, exploration algorithms can be categorized into two classes: deliberative and reactive. Deliberative exploration may guarantee the completeness of coverage. However, to build the map of the environment, the robots need to have accurate sensors and location information, explicit and rapid communications, and powerful computation capabilities. For the application scenario and surveillance system in this study, these requirements are not very affordable or realistic.

Reactive exploration usually depends on the robots' individual sensing. Typical exploration algorithms without mapping include artificial potential field-based exploration, visibility-based exploration, etc. These algorithms are simple, easily deployable and scalable to large groups of mobile and static nodes, but are unable to guarantee full coverage. This study therefore focuses on reactive exploration algorithms, and tries to find methodologies (that include heuristics) to maximize the coverage of the environment.

In this thesis, three exploration algorithms are proposed for multiple robots to search for targets in unknown environments cooperatively:

- Potential field-based exploration
- Swarm intelligence exploration
- Landmark-based exploration.

4.1.1 Potential Field-based Exploration

Potential field-based exploration is a well-known exploration strategy that has been studied and used in many applications because of its simplicity and scalability. The idea of potential field-based exploration is to regard the obstacles and neighbor robots as repulsive force sources and let the robots move under these forces, such that the robots can be dispersed throughout the whole area (Howard et al., 2002). However, if there are no repulsive forces (no obstacles nearby), or the repulsive forces are balanced (in equilibrium states), the robots will not move to explore the environment further. A solution is to add a random attractive force to the robots to trigger them to move (usually randomly) to extend the coverage of the environment. The concept of potential field-based exploration is depicted in Figure 4-1.

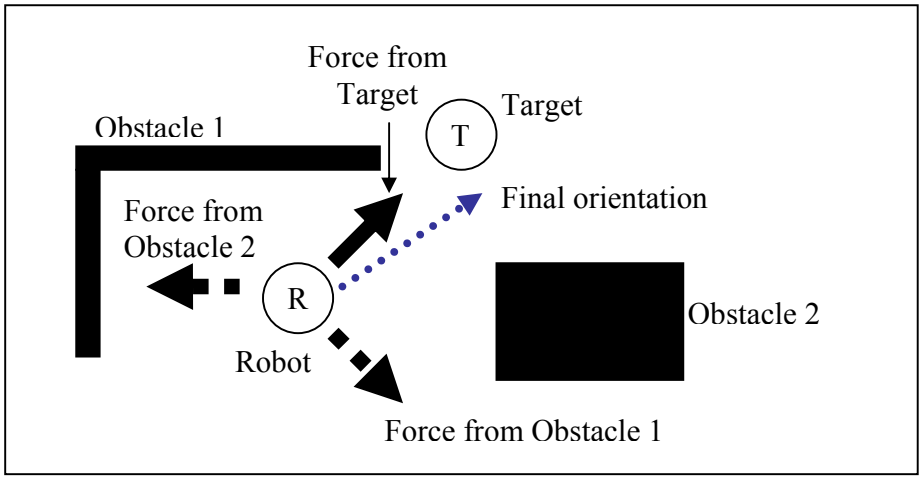

Figure 4-1 Potential Field-based Exploration.
The robot receives two virtual repulsive forces (dashed arrows) from obstacles 1 and 2, and one attractive force (solid arrow) from the target location. Finally, the robot will move along the orientation of the summation of these forces (slim dashed arrow).

In this thesis, the potential field-based exploration algorithm is selected as the benchmark for comparison purposes. Therefore, it is important to know the implementation details of this

method. Potential field-based exploration typically follows the procedures introduced in Figure 4-2.

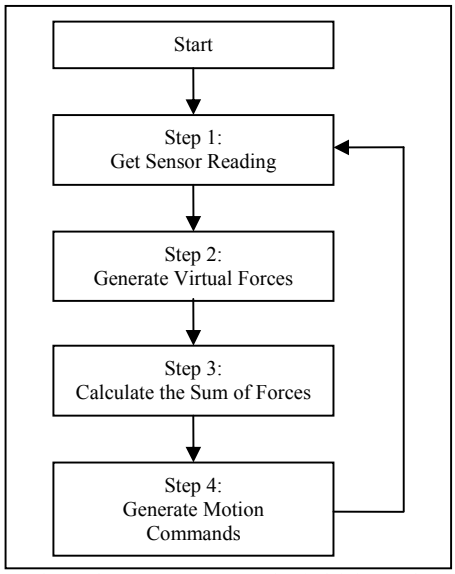

Figure 4-2 Flowchart of Potential Field-based Exploration

In step 1, the robot uses its sensors to get information about the environment. As shown in Figure 4-3, the robot detects its surroundings using sonar sensors. The robot has 64 sonar sensors around it with even separation. Each sonar sensor can detect the distance to the object along the orientation of the sensor.

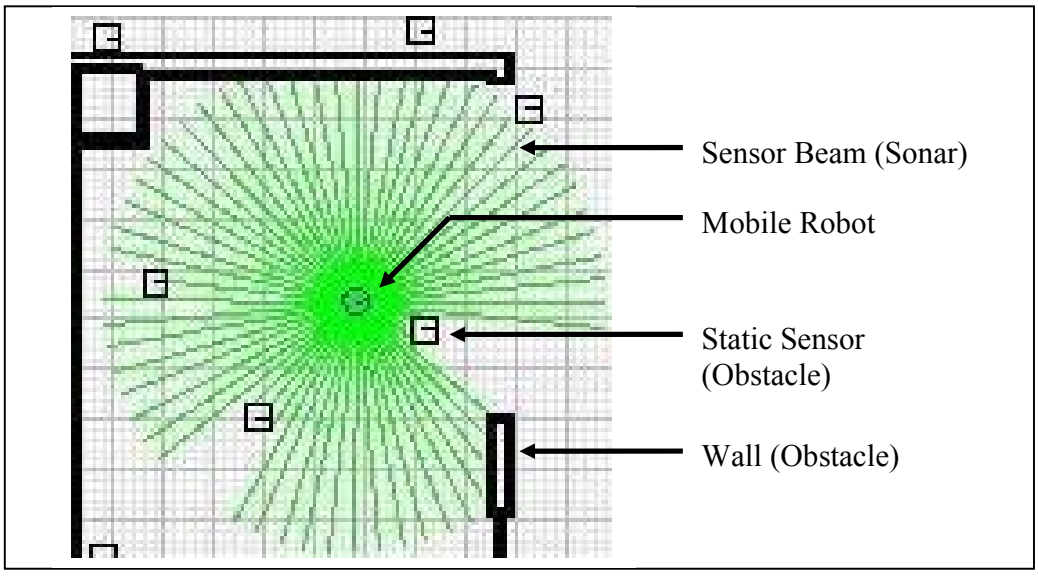

Figure 4-3 Sensor Reading of Robot

In step 2, the robot translates the sensor readings into virtual forces. With respect to the robot shown in Figure 4-3, there are 64 proximity sensor readings about its surrounding. If a proximity sensor detects something within its sensing range, there is a virtual repulsive force along this orientation of the sensor (towards the robot). The magnitude of the repulsive force is defined in Figure 4-4 (the maximum magnitude value is normalized as 1). Let r_{r-o} be the distance between robot and the object. The magnitude of the repulsive force depends on r_{r-o} according to:

- $r_{r-o} \leq r_{r1}$: The magnitude is set as maximum. In this segment, the repulsive force is strongest; hence the robot will leave the obstacle rapidly.
- $r_{r1} < r_{r-o} \leq r_{r2}$: The magnitude decreases gradually to zero; hence the repulsive force will decrease continuously.
- $r_{r-o} > r_{r2}$: In this segment, the magnitude is set to zero because the robot is sufficiently apart from the obstacle. There is no repulsive force to the robot.

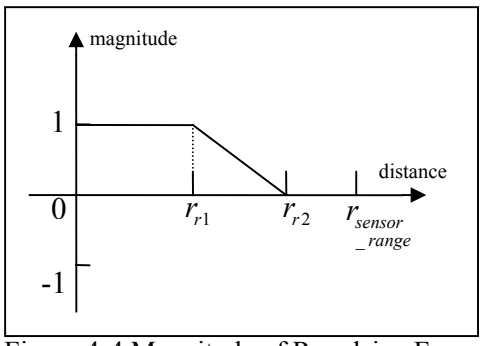

Figure 4-4 Magnitude of Repulsive Force

In step 3, the robot generates an attractive force (toward the desired destination, or a random selected direction), and then calculates the summation of the virtual forces (attractive forces and repulsive forces) using vector operations. This is an important step for the potential field-based exploration.

In step 4, the robot generates the motion commands according to the summation of the virtual forces. For the differential wheel robots (as shown in Figure 4-5), two commands for the left wheel and right wheel are required to control the motion.

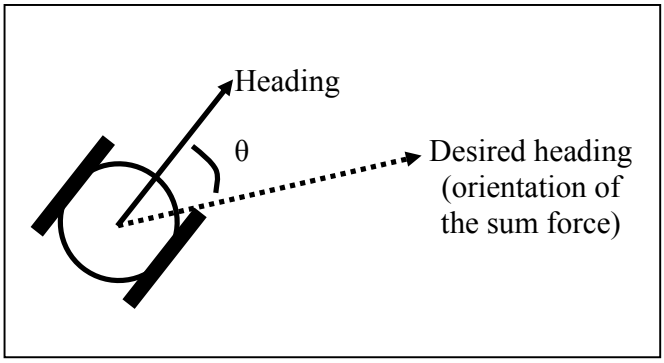

Figure 4-5 Generation of Motion Commands for Differential Wheel Robot
(Angle θ is the difference between the current heading and desired heading of the robot)

To let the robot move along desired orientation at proper speed, the wheel speeds can be generated using the following Equation (4.1):

$$\begin{aligned} \text{Left wheel speed} &= \left|\vec{F}\right| * w_v + \theta * w_a \\ \text{Right wheel speed} &= \left|\vec{F}\right| * w_v - \theta * w_a \end{aligned} \tag{4.1}$$

In this Equation (4.1), $\vec{F}$ is the summation of the virtual forces, including attractive forces and repulsive forces, as introduced in Figure 4-1; θ is the difference between the current heading of the robot and the orientation of $\vec{F}$; w_v and w_a are the appropriate weights to generate suitable commands for the robot. According to this equation, if the robot is driven by strong forces (large $\left|\vec{F}\right|$), its two wheels will turn faster; if the robot has large bias from desired orientation (large θ), its left and right wheels will have large and opposite turning to adjust.

The entire process of the potential field-based exploration that is implemented in this thesis is shown in Algorithm 4.1.

Algorithm 4.1. Potential Field-based Exploration (for Robot *i*)

Step 1. Set initial force to move as $\vec{F}$. The orientation of $\vec{F}$ is a random value that is uniformly distributed between [0, 2π); the magnitude of $\vec{F}$ is a fixed value, e.g., 1.0.

Step 2. Scan the surrounding environment; find the sets of detected objects, *Obj*. This set *Obj* can also be the set of sensor readings (as shown in Figure 4-3).

Step 3. If *Obj* is not empty, for all $O_j \in Obj$, let $\vec{F} = \vec{F} + \vec{O}_j$;

Here, $\vec{O}_j$ is the repulsive force between robot *i* and object *j*. The orientation is from *j* to *i*; the magnitude is usually calculated using the following Equation (4.2):

$$\left|\vec{O}_j\right| = \begin{cases} 1, & \text{if } d_{ij} \le r_{r1} \\ \dfrac{r_{r2} - d_{ij}}{r_{r2} - r_{r1}} - 1, & \text{if } r_{r1} < d_{ij} \le r_{r2} \\ 0, & \text{if } d_{ij} > r_{r2} \end{cases} \tag{4.2}$$

In this equation, d_{ij} is the distance between the center of robot *i* and nearest point of object *j*, and r_{r1} and r_{r2} are positive constants.

Step 4. Let robot *i* move under the virtual force $\vec{F}$.

Step 5. Goto Step 1.

Since the proposed potential field-based search only relies on the local sensing of the robot, it is totally distributed and scalable for any number of robots. However, due to the lack of

communications among robots, and the absence of memory of covered areas, the cooperation in exploration is at a low level. Consequently, the robots may repeatedly explore the same area, resulting in poor coverage of the entire environment.

In this thesis, the Artificial Potential Field (APF)-based exploration algorithm (noted as "random" exploration) is used as the lower-bound benchmark for comparing the exploration and target-searching algorithms.

4.1.2 Swarm Intelligence Exploration

Swarm intelligence exploration is proposed as an enhancement to potential field-based exploration. In this approach, neighboring robots move along different orientations; hence they can cover different regions of the environment. Intuitively, such swarm intelligence can increase the coverage of the environment as compared to potential field-based exploration. Details of the methodology are as follows: when a robot i is moving, it will periodically broadcast its ID (i) and heading (H_i) to its neighbors. This is a one-hop broadcast that is only heard by its neighbors. At the same time, when a robot is moving, it also listens to the wireless channel continuously so that it can receive the broadcasts from its neighbors. If it receives a message from any of its neighbors, it may modify its heading accordingly. The implementation of swarm intelligence exploration comprises two processes: (i) modification of the moving orientation of the robot (Algorithm 4.2); and (ii) periodic update of the desired orientation according to information gathered from neighboring robots (Algorithm 4.3).

Algorithm 4.2. Swarm Intelligence Exploration – Modification of the Moving Orientation (for Robot i)

Step 1. Set initial force to move as $\vec{F}$. The orientation of $\vec{F}$ is a random value that is uniformly distributed between [0, 2π); the magnitude of $\vec{F}$ is a fixed value, e.g., 1.0.

Step 2. Check the heading information (H_i) obtained from the heading updates (Algorithm 4.3). Set the orientation of $\vec{F}$ to be equal to H_i. H_i is the absolute value of the orientation. The range is 0 to 360 degrees (0 at North, increasing clock-wisely, viewed from above).

Step 3. Scan the surrounding environment; find the sets of detected objects, *Obj*.

Step 4. If *Obj* is not empty, for all $O_j \in Obj$, let $\vec{F} = \vec{F} + \vec{O}_j$;

Here, $\vec{O}_j$ is the repulsive force between robot *i* and object *j*. The orientation is from *j* to *i*; and the magnitude is usually calculated by Equation (4.2).

Step 5. Let robot *i* move under the virtual force $\vec{F}$.

Step 6. Goto Step 1.

Algorithm 4.3. Swarm Intelligence Exploration – Heading Updates (for Robot *i*)

Step 1. Set initial values: $R_{neighbor} = \infty$, H_i = current heading of robot *i*.

Step 2. Broadcast H_i and ID (*i*) to neighboring robots <*i*, H_i>.

Step 3. Listen constantly for a period of time, during which,

if a message <*j*, H_j> is received from neighbor robot *j*

if (*j* > *i*) and (*j* < $R_{neighbor}$)

let $R_{neighbor}$ =*j*, $H_{neighbor} = H_j$

Step 4. If $R_{neighbor} \neq \infty$, let $H_i = H_{neighbor}$ + 90 degree

Step 5. If H_i >360, let $H_i = H_i$ - 360. Pass information about H_i to the process in Algorithm 4.2.

Step 6. Goto Step 1.

According to the above algorithms, a robot will change its heading only if the ID of its neighboring robot is higher. If a robot has more than one neighbor with higher IDs, it will change its heading with respect to the robot with the closest ID that is higher than its own ID. Figure 4-6 depicts how this algorithm works. Initially, the four robots move along randomly selected directions. After exchanging their heading information, robot 28 keeps its previous heading because its ID is the highest, while robots 27, 26, and 25 change their headings according to robots 28, 27, and 26, respectively. Finally these robots will move along different directions with certain difference in their headings.

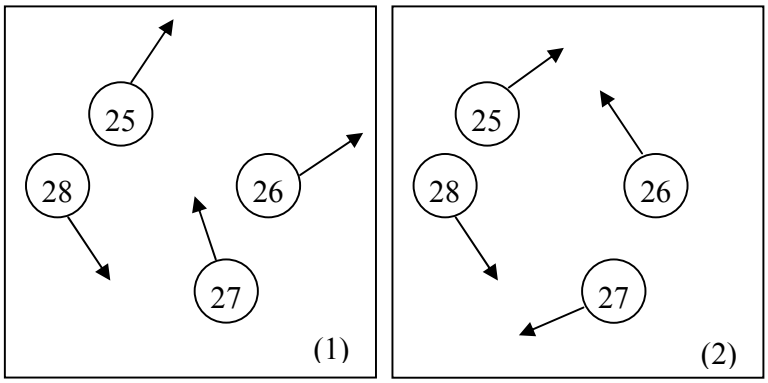

Figure 4-6 Swarm Intelligence Exploration
(1). Initial Orientations; (2). Adjusted Orientations after Negotiation

4.1.3 Landmark-based Exploration

Potential field-based exploration (Section 4.1.1) and swarm intelligence exploration (Section 4.1.2) can disperse the robots in the environment to achieve better coverage. However, they cannot guarantee full coverage because none of them has memory of the covered area. To increase the coverage more effectively, the robots have to move towards the uncovered regions. Using cooperation between mobile robots and static sensors, the robots can avoid the re-exploration of covered regions and discover uncovered regions without having to build a map of the area. The idea works by letting some static sensors serve as landmarks to remember the visit

history of their neighborhoods. This information is then passed on to newly arriving robots to help them decide on the exploration region. The implementation of landmark-based exploration comprises two processes: (i) the continuous change in motion of the robot (Algorithm 4.4); and (ii) the periodic update of the desired orientation according to information obtained from the landmarks (Algorithm 4.5).

Algorithm 4.4 Landmark-based Exploration - Motion Decision (for Robot *i*)

Step 1. Set initial force to move as $\vec{F}$. The orientation of $\vec{F}$ is a random value that is uniformly distributed between [0, 2π); the magnitude of $\vec{F}$ is a fixed value, e.g., 1.0.

Step 2. Check the heading information (H_i) obtained from the heading updates (Algorithm 4.5). Set the orientation of $\vec{F}$ to be equal to H_i. H_i is the absolute value of the orientation. The range is 0 to 360 degrees (0 at North, increasing clock-wisely, viewed from above).

Step 3. Scan the surrounding environment; find the sets of detected objects, *Obj*.

Step 4. If *Obj* is not empty, for all $O_j \in Obj$, let $\vec{F} = \vec{F} + \vec{O}_j$;

Here, $\vec{O}_j$ is the repulsive force between robot *i* and object *j*. The orientation is from *j* to *i*; the magnitude is usually calculated by Equation (4.2).

Step 5. Let robot *i* move under the virtual force $\vec{F}$.

Step 6. Goto Step 1.

Algorithm 4.5 Landmark-based Exploration - Heading Updates (for Robot *i*)

Step 1. Set initial values: H_i = current heading of robot *i*.

Step 2. Listen constantly for a period of time, during which,

if a message is heard from a neighboring landmark static sensor *j*: <*j*, v_1, v_2, v_3, v_4>

(v_*x* is the number of visits in the sub-regions around this landmark *j*),

send the relative orientation (the orientation from robot to landmark) to static sensor *j*: <*i,* orientation>

Step 3. Find the smallest value of v_*x* from {v_1, v_2, v_3, v_4}, let H_i = heading towards region v_*x* of landmark *j*.

Step 4. Pass information about H_i to the process in Algorithm 4.4.

Step 5. Goto Step 1.

It should be noted that the static sensor (landmark node) computes the relative location of the robot from <*i,* orientation> (sent in Algorithm 4.5) and updates this information in its own memory. The static sensors (landmark nodes) will then periodically broadcast the information stored in their memory to its surroundings.

Figure 4-7 shows an example of how landmark-based exploration works. Around the landmark node S5, the region is divided into 4 sub-regions: north, west, east and south. The memory of S5 maintains the number of times that sub-region has been visited by a robot.

- Figure 4-7-(1): The counters of landmark node S5 are all set to 0 initially, indicating that no robot has visited any of the regions.
- Figure 4-7-(2): S5 broadcasts its memory (counters for all regions indicating the neighborhood visit history) as <landmark 5, N0, W0, E0, S0>. Robot 25 receives this message. Because all four sub-regions around S5 have not been visited, Robot 25 decides to move randomly. In this example, it moves to the west region of node 5.
- Figure 4-7-(3): Robot 25 detects its relative angle to node 5, and sends this information to S5. S5 now learns that Robot 25 is in its south and west regions and increments the counter for these regions by 1.

- Figure 4-7-(4): Robot 25 leaves the west region of S5. S5 updates and keeps the visit history.
- Figure 4-7-(5): S5 broadcasts its memory as <landmark 5, N0, W1, E0, S1>. Robot 26 receives this message and then decided to move to the east region of node 5. Robot 26 updates S5, which increments the counter of its south and east regions by 1.
- Figure 4-7-(6): Robot 25 leaves the east region of S5. S5 updates and keeps the visit history.

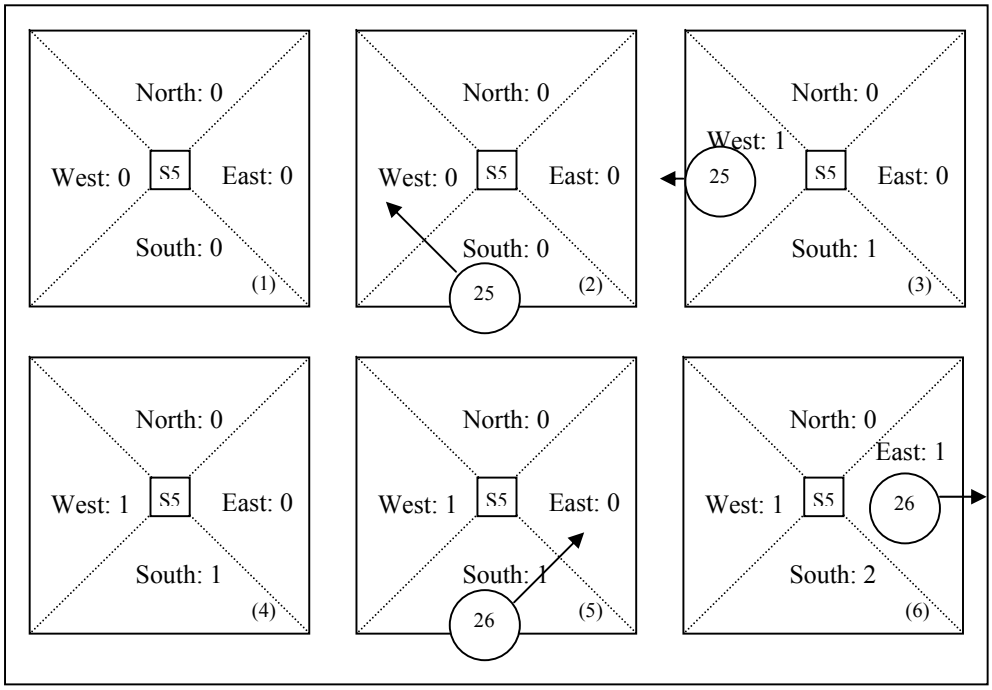

Figure 4-7 Landmark-based Exploration

If there are sufficient landmark nodes deployed throughout the environment, the proposed landmark-based exploration can optimize the distribution of robots in the environment and finally achieve full coverage of the entire area. However, if all static nodes are assigned as landmarks, the communication overhead will be excessively high, especially in areas with many sensors. Therefore, only nodes that are located at critical places, e.g., the entrance of a room, are selected to be landmarks. This selection can be done randomly or deliberately by the static nodes themselves through the following mechanisms:

- Self-discovery of the vicinity. The static sensor may carry specific sensor, e.g., camera, to detect the surroundings and then decide to serve as a landmark node to help robots.
- Self-discovery of the number of neighbors. In proposed application scenarios, the initial deployment of static sensors may be by throwing-in. In this case, the static sensors with fewer neighbors might be at the boundary of cluster of sensors. Such sensors could volunteer to be the landmark nodes.
- Self-discovery of the energy level of batteries. For static sensors, saving energy is of great importance. Because communications may cost a lot of power, the static sensor with high battery level could become the landmark node.

4.1.4 Summary

The methodologies, assumptions, strengths and weaknesses of the three proposed exploration algorithms are summarized in Table 4-1.

Table 4-1 Comparison of Proposed Exploration Algorithms

Algorithm	Methodology	Assumptions	Pros	Cons
Potential field-based exploration	All mobile robots search along random orientations unless obstacles change orientations.	None.	Simple, scalable.	Inefficient; cannot guarantee 100% coverage.
Swarm intelligence exploration	Neighboring robots share their headings to move along different orientations, unless obstacles change orientations.	Compass; Communication among mobile robots.	Simple, scalable, efficient.	Cannot guarantee 100% coverage.
Landmark-based exploration	Inter-mobile robot and static sensor communication enables mobile robots to search in different areas in the environment.	Communication between mobile robots and static sensors; Compass; Ability to distinguish neighboring sensors.	Efficient, can achieve full coverage if there are sufficient landmarks	Large communication overhead.

These exploration algorithms can be used to search for both embodied targets and virtual targets because they allow the robots to cover the environment efficiently.

4.2 Searching

To find desired targets, the exploration algorithms may use the brute-force search methodology, which attempts to increase the probability of finding targets by maximizing the coverage of the environment. On the other hand, searching algorithms could try to increase the probability of finding targets by relying on clues that are related to the targets. As introduced in Section 3.4, the environment has both embodied and virtual targets to be found. In this subsection, one target-centric searching algorithm is proposed to find the virtual targets, i.e., the communication gaps in the ad hoc communication networks, by following the clues about the targets.

4.2.1 Hop-Count Gradient-orientated Searching

In the proposed application scenario, there are four sinks (beacons) at corners. The static sensors can calculate, update, and store the hop-count information with regard to each sink. This hop-count information indicates the distance from the sensors to the sinks: a node that is further away from a sink will typically have larger hop-counts to this sink because the data packets from this sensor have to be relayed by more intermediate sensors to reach this sink. The hop-count information is used for the hop-count-based localization (to be introduced in Chapter 6), and can also be used to detect and locate communication gaps. Usually, sensors that are further away from a sink are more likely to be at the boundary of the cluster (group) of connected sensors (to this sink); therefore, the hop-count information can help indicate the communication gaps. As shown

in Figure 4-8, the disconnected sensor cannot communicate with either *Sink 1* or *Sink 2* because it is too far away from the sensors that are connected to these sinks. To enable connectivity between the isolated sensor and the sinks, the robot needs to find and bridge the critical areas (communication gaps) which are at the further boundary (with respect to the sink) of the group of connected sensors.

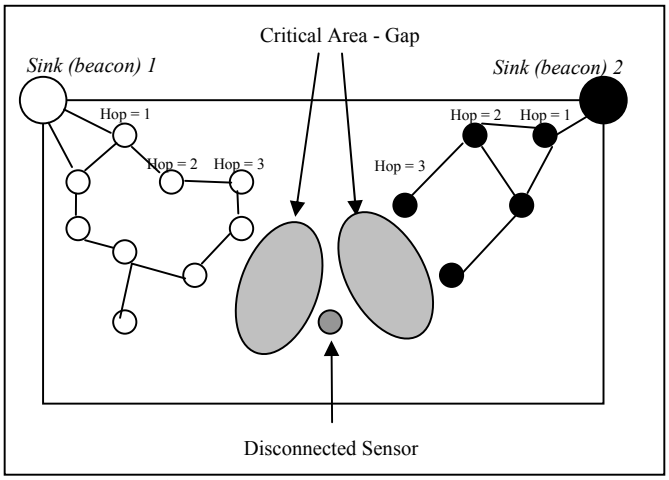

Figure 4-8 Critical Areas in the Ad Hoc Communication Network

To enable a robot to move along the direction of increasing hop-counts, the robot is required to know the hop-counts and ID of its neighboring static sensors. When the robot moves, it will continuously request and receive hop-count information from its neighboring sensors. It will then move along the direction where the hop-counts increase the most. As shown in Figure 4-9, the robot can hear the hop-count information of five neighbors (sensors 1, 2, 3, 4, 6) within its communication range. It will then move along the direction which is parallel to the link from sensor 1 to 6. This is because the maximal hop-count increase is from sensor 1 to sensor 6 (which are 1 hop and 4 hops away from *Sink 1*, respectively).

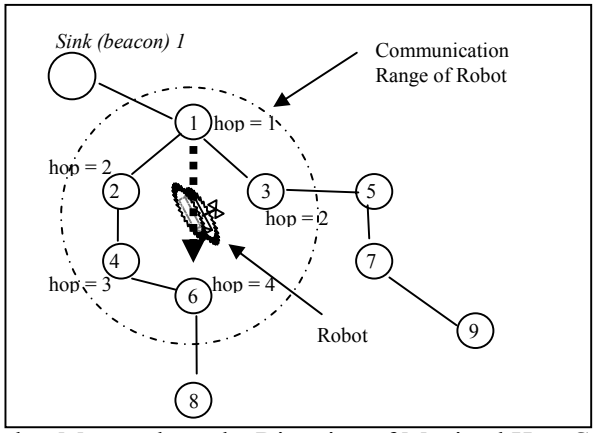

Figure 4-9 Robot Moves along the Direction of Maximal Hop-Count Increase

The connected sensors (i.e., any sensor that is connected to at least one sink) can help the robot to select the appropriate orientation to move towards areas where the gaps are likely to appear. In other words, the robots can utilize the hop-count information to search for the communication gaps (virtual targets) in the promising areas.

There are two main problems associated with the hop-count gradient-oriented searching, namely the local minima problem and conflicting gradient problem.

The local minima problem occurs when nearby robots follow the same path because their neighboring sensors are the same. This is not efficient because the robots are working like one robot, which is a waste of robot resources. To alleviate this problem, swarm intelligence (proposed in Section 4.1.2) is used to modify the desired orientation of the robots, resulting in:

$$Heading = w * \alpha_{hop\_count\_gradient} + (1\text{-}w) * \alpha_{swarm\_intelligence} \quad (4.3)$$

The angle calculated for each robot that is making use of the swarm intelligence exploration (Algorithm 4.3) will separate the paths of the robots. In Equation (4.3), *Heading* is the desired orientation of the robot; $\alpha_{hop\_count\_gradient}$ is the orientation obtained from the hop-count gradient;

$\alpha_{swarm\_intelligence}$ is the swarm intelligence orientation; and w ($0 \le w \le 1$) is the weight used to adjust the influence of swarm intelligence. If w is large, the robot is more likely to move along the hop-count gradient; otherwise it is more likely to move apart from the neighboring robots.

In the proposed surveillance scenario, it is possible that a robot receives conflicting hop-count gradients from the neighboring sensors. For example, a robot may have two neighbors, sensors *A* (hop-counts: 3, 8, 5, 5) and *B* (hop-counts: 4, 7, 5, 5). In this case, the hop-count with respect to beacon 1 increases from *A* (3) to *B* (4), but the hop-count with respect to beacon 2 increases from *B* (7) to *A* (8). To avoid such conflicting hop-count values, an additional rule is imposed such that the robot will only move along the direction where the hop-counts (with respect to all beacons) increase. If there is no such direction, the robot will not follow the hop-count gradient of any beacon. In this case, the behavior of robots is the same as the swarm intelligence exploration.

4.2.2 Summary

In the proposed ad hoc sensor network, the communication gaps usually exist at the further parts of the sensor group (with regard to the reference/beacon nodes); therefore, the robots may have more chances to find the gaps at such promising areas. The hop-count gradient-oriented searching algorithms can enable the robots to search at promising areas by detecting the hop-count information from their neighbor sensors. This algorithm should be efficient for the proposed application scenario. On the other hand, this algorithm requires communications between mobile robots and static sensors, and the robots should have the ability to detect the ID of neighboring static sensors.

4.3 Simulation Tests and Discussions

To demonstrate the efficacy of the proposed exploration and target-searching algorithms, simulation tests are conducted to evaluate the performance of each algorithm. Because the three exploration algorithms are proposed for searching both embodied and virtual targets in unknown environments, the following metrics are used to evaluate their efficacy:

- Average number of embodied targets found (*Ave_T*)
- Average time spent to find the embodied targets (*Ave_L*)
- Total number of static sensors that are connected (*Total_Conn*) - the number of sensors that are connected to at least one of the four sinks. This is a measure of the improvement of network connectivity after the communication gaps have been found and bridged.
- *k*-connectivity of static sensors (*k_Conn*) - The average number of sinks that each static sensor is connected to, where $0 \leq k \leq n$ (n=4 is used in this study). This is another measure of the improvement of network connectivity after communication gaps have been found and bridged. This metric is especially meaningful for the hop-count-based localization – to localize a node in a 2D environment, the *K_Conn* should be equal or larger than 3.

With respect to the hop-count gradient-oriented target-searching algorithm, because it is specially designed to search for the virtual targets (communication gaps) in unknown environments, the following metrics are used to evaluate its efficacy:

- Total number of static sensors that are connected (*Total_Conn*)
- *k*-connectivity of static sensors (*k_Conn*).

In these metrics, the *Ave_T* and *Ave_L* are related to the embodied targets (such as the intruder or special objects in the environment). *Ave_T* is the number of targets that are found by the robots –

the higher the value of *Ave_T*, the higher the efficiency of the algorithm. *Ave_L* is the average length of time taken by the robots to find a target – the lower the value of *Ave_L*, the higher the efficiency of the algorithm.

The *Total_Conn* and *K_Conn* are related to the virtual targets (such as the communication gaps). When a robot finds such targets, it will deploy new sensors to enable communication between the disconnected sensors. *Total_Conn* is the number of sensors that are connected to at least one of the four sinks. With higher connectivity, more sensors are able to communicate with the sink(s). *K_Conn* is the average number of sinks that each static sensor is connected to, where $0 \leq k \leq n$ (n=4 is used in this study). With higher *K_Conn*, the sensor nodes can multi-cast the data packets to more than one sink to increase the communication reliability (Seah & Tan, 2004). In addition, in hop-count-based localization (Wong, et al., 2005), the *K_Conn* is important because the sensors need to have at least k=3 (for a 2D scenario) to perform triangulation to estimate the locations. Furthermore, the localization accuracy improves with increasing *K_Conn* values.

4.3.1 Simulation Environment and Settings

As compared to conventional exploration and target-searching algorithms, the techniques proposed in this study are more practical because more realistic assumptions are made. To reflect the real scenario, the simulation environment is set as the following:

- The environment and robot is simulated by Player/Stage, a well-known robotics simulator (Gerkey et al., 2003).
- In the proposed surveillance system, the communication is critical because it affects the cooperation among robots and static sensors, and influences the message delivery. While traditional robotics research usually assumes ideal communications with no

communication delays or failures, this study makes use of GloMoSim (Zeng et al., 1998), a networking simulator, to simulate the real communication processes with realistic limitations and constraints.

- The time synchronization and event signaling are achieved between Player/Stage and GloMoSim using semaphores and shared memory.

4.3.2 Simulation Results and Discussion

4.3.2.1 Embodied Targets

One of the key requirements of a surveillance system is to search for the embodied targets. In this study, the proposed exploration and target-searching algorithms are applied and tested in a representative indoor environment that has been used by many robotics researchers, shown in Figure 3-1. The environment has dimensions 57m by 44m and a diagonal of 72m. Four beacons are deployed at the corners of this area. Total 50 static sensor nodes are randomly deployed during the initialization phase before the 10 mobile robots enter the area (5 from the left and 5 from the right side of the environment). The mobile robots can move at a maximum speed of 1.2m/s and have a maximal turning rate of 144deg/s. In this environment, three targets are randomly placed in the surveillance area. Both static sensors and mobile robots have a communication range of 8.865m and use a simplified AODV (Perkins & Royer, 1999) protocol to relay packets to the control center through beacons (at corners). The mobile robots make use of proximity sensors, e.g., sonar sensors, to detect obstacles. The sensing range is approximately 4m. Each static sensor periodically broadcasts its hop-count information to its neighborhood. Mobile

robots, newly deployed sensors and disconnected sensors can request hop-count updates from their neighbors, which will then reply with their hop-count information immediately.

With regard to each exploration and searching algorithm, 10 simulation runs are conducted, and each run lasts 12 minutes. The average number of embodied targets found (*Ave_T*) in all runs is shown in Figure 4-10. Examining this figure, we find that the landmark-based exploration algorithm outperforms the swarm intelligence and potential field-based (random) exploration algorithms. During the simulation, the potential field-based (random) exploration cannot effectively disperse the robots, such that the robots may search in the same region. This is a waste of robot forces and therefore degrades the system performance. Swarm intelligence exploration lets the robots move along different orientations. Therefore, the robots are more likely to search in different regions and their coverage of the environment is larger than potential field-based exploration. As a result, more targets are found. Landmark-based exploration is most efficient because it not only separates the robots using different headings, it also enables robot-landmark cooperation to lead the robots to search in uncovered areas. If there are sufficient landmarks deployed all over the environment, the landmarks can help the robots explore the entire environment. Therefore, landmark-based exploration achieves the highest performance in target searching.

In addition to the absolute value of *Ave_T*, the standard deviation of *Ave_T* for different simulation runs is shown in Figure 4-11. This figure shows that the landmark-based exploration algorithm has more consistent results: in different simulation runs, it can achieve very similar performance (small standard deviation), while the performances of swarm intelligence and potential field-based (random) exploration vary more significantly. This result shows that the memory stored in the landmark sensors can optimally allocate the mobile robots to search in different areas; therefore the routine of the mobile robots is more deterministic than that of the

other two exploration algorithms. Swarm intelligence exploration can let the robots choose different orientations, therefore the routine of the robots is more deterministic than that of potential field-based exploration, which is totally random.

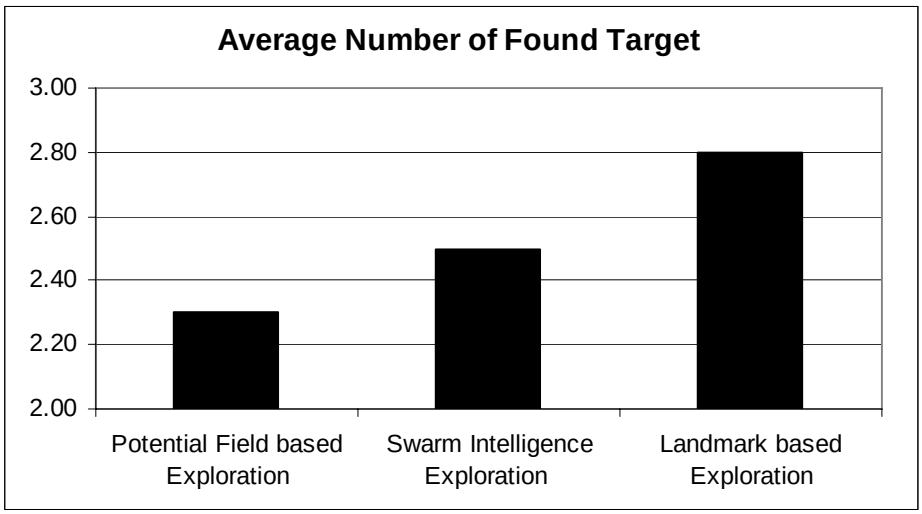

Figure 4-10 Average Number of Targets Found

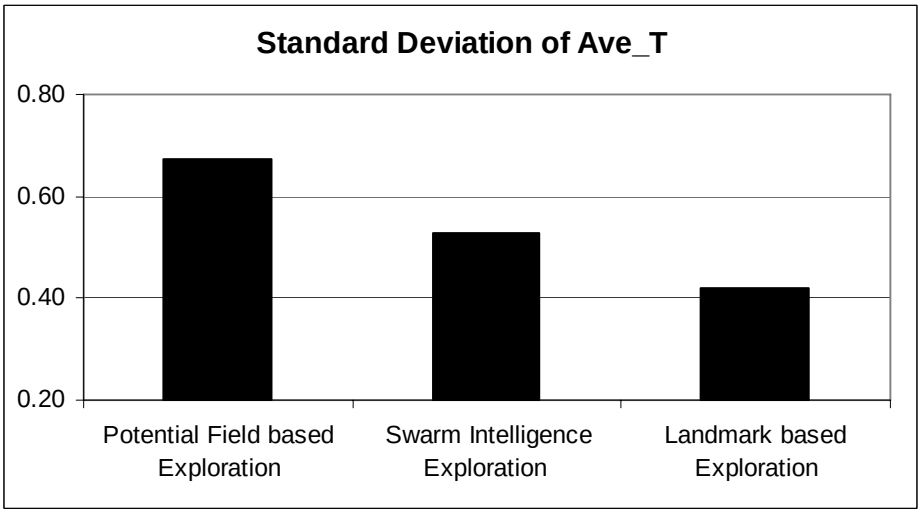

Figure 4-11 Standard Deviation of *Ave_T*

Besides the number of targets found, the average searching time per target (*Ave_L*) is also of great importance when evaluating the efficacy of the exploration algorithms. Figure 4-12 compares the *Ave_L* of different exploration algorithms. In this figure, the value of *Ave_L* is normalized by setting the value of *Ave_L* in the potential field-based exploration to be "1". From

Figure 4-12, we can see that the landmark-based exploration algorithm outperforms the swarm intelligence and potential field-based (random) exploration algorithms. This is consistent with the comparisons made with *Ave_T*.

The standard deviation of *Ave_L* for different simulation runs is shown in Figure 4-13. In this figure, the value of the standard deviation is normalized by setting the value of *Ave_L* in the potential field-based exploration to be "1". Similar to Figure 4-11, this figure also shows that the performance results of the landmark-based exploration algorithm are more consistent.

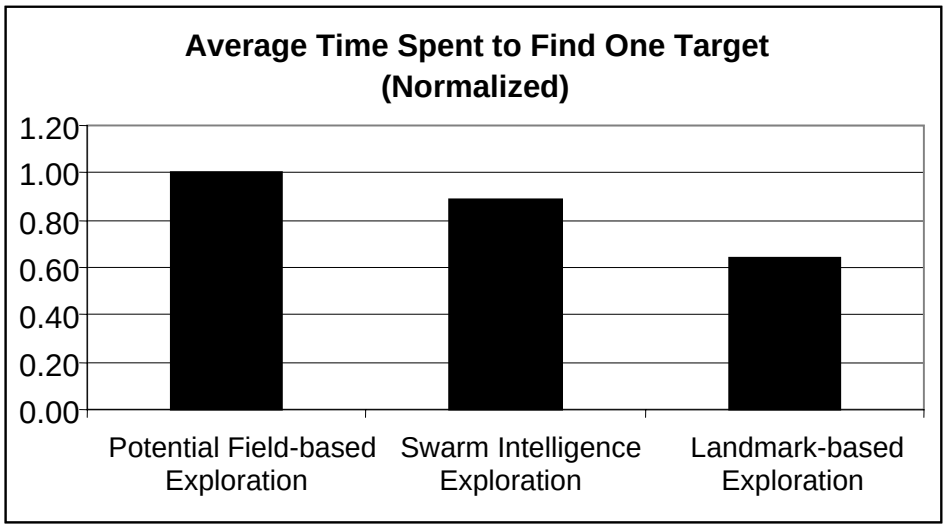

Figure 4-12 Average Time Spent to Find One Target (Normalized)

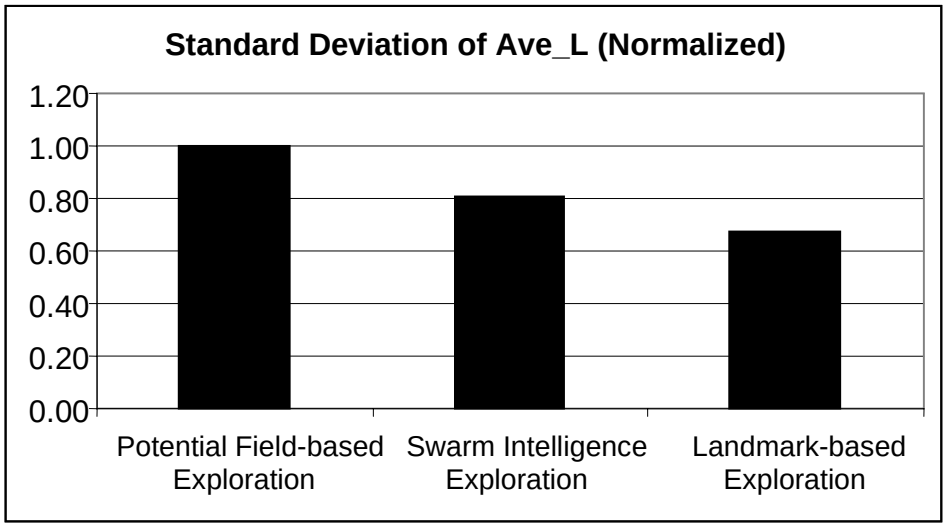

Figure 4-13 Standard Deviation of *Ave_L* (Normalized)

Videos and photos have been taken for above simulations. Please kindly refer to http://guppy.mpe.nus.edu.sg/~mpeangh/kevin to review them to have a better understanding of the system.

4.3.2.2 Virtual Targets (Communication Gaps)

The proposed surveillance system, which includes both mobile robots and static sensors nodes, can be efficient and economical for real applications. However, this system requires reliable network connectivity for the purpose of communications. Therefore, it is important that the mobile robots search and bridge the communication gaps in the environment.

To evaluate the performance of the proposed algorithms when searching for virtual targets (communication gaps), the previous simulation environments (Figure 3-1) are not very suitable due to the following considerations:

- The network shown in Figure 3-1 is not sufficiently large. If the environment is too small, the ad hoc network communication technologies are not meaningful because the nodes may communicate directly. Furthermore, to test and implement ad hoc communication-based algorithms (e.g., hop-count-based exploration), it might be better if there are many nodes to cooperate with robots. Therefore, the size of the environment and the number of nodes should be large.
- In real environments, the wireless signals can pass through walls and obstacles; however, the robots are unable to pass through these objects. Due to this reason, the exploration and target-searching algorithms (e.g., swarm intelligence exploration, landmark-based exploration, and hop-count-based searching) may not work properly in environments

with large internal obstacles. For example, hop-count gradient-oriented searching may lead the robot to move along a direction toward the wall; however, the robot cannot cross the wall as the wireless signal does.

Based on above considerations, the proposed exploration and target-searching algorithms are applied and tested in a large environment with small obstacles, in this study. The environment under consideration is shown in Figure 4-14. Because there are only small obstacles in this environment, we may concentrate on the cooperation between mobile robots and static sensors, and thus make fair comparison among the proposed exploration and target-searching algorithms. After this, in future we may extend the simulation tests to more complex scenarios with large internal obstacles. For such environments, the exploration and target-searching algorithms need to be modified to handle large obstacles; therefore the performance comparison is for both "exploration and searching" and "obstacle avoidance".

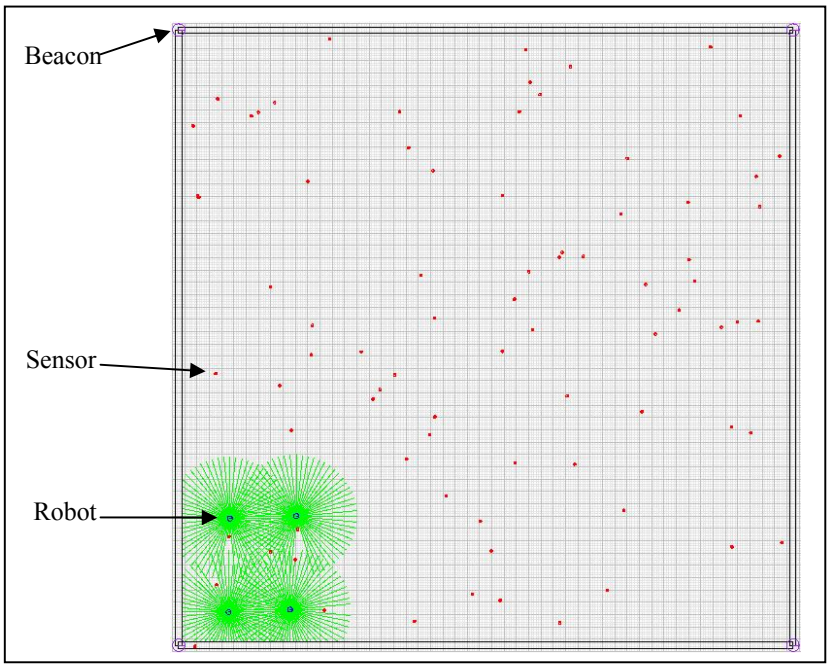

Figure 4-14 Simulation Environment for Exploration and Searching of Virtual Targets (Communication Gaps)

In this simulation environment, a total of 80 static sensors are randomly deployed in the 2.5km by 2.5km area during the initialization phase. A reference node (acting as the beacon/sink) is placed at each of the four corners. Four robots explore the environment and drop up to a maximum of 20 new sensors in the environment to improve network connectivity. The transmission range of the wireless transceiver is 300 meters; the speed of the robots is 2 meters/second (in forward motion) and 3 degrees/second (in turning).

For each exploration and target-searching algorithm, 10 simulation runs are conducted, and each simulation run last 200 minutes. In each run, the 80 pre-deployed static sensors start work in the beginning (time zero). It takes about 10 to 15 minutes for the sensors to construct the ad hoc networks, i.e., each node discover its neighbors and find the link (either single-hop or multi-hop) to the sinks if it could. In the no-mobility mode, the extra 20 static sensors are thrown in at time zero, and they communicate together with the 80 pre-deployed sensors. In other modes, the robots start moving at time 15 minutes, after the ad hoc network has been constructed. It will take some time for the robots to find the communication gap (virtual target). When a robot find a communication gap by the mechanism introduced in Section 3.4.2, it will deploy a new static sensor at that location. This newly deployed static sensor will start communicating and improve the network connectivity. It should be noted that in this approach, the communications among robots and sensors is simulated by GloMoSim, a communication simulator. GloMoSim could simulate the fluctuation of wireless signal, the noise in communication channel, and the interference between transceivers.

Since the purpose of finding the virtual targets (communication gaps) is to improve the quality of the network communications, the metrics relating to connectivity (*Total_Conn* and *k_Conn*) are used to measure the performance of exploration and searching.

Figure 4-15 and Figure 4-16 show the connectivity enhancements by different exploration and searching algorithms. Under most circumstances, the performance of the exploration and target-searching algorithms increases in the following order: (i) potential field-based (random) exploration; (ii) swarm intelligence exploration; (iii) landmark-based exploration; and (iv) hop-count gradient-oriented target searching. In these figures, a "no mobility" model is also added to illustrate the performance improvement in the connectivity if no robots are sent into the environment and the 20 new sensors are randomly thrown in.

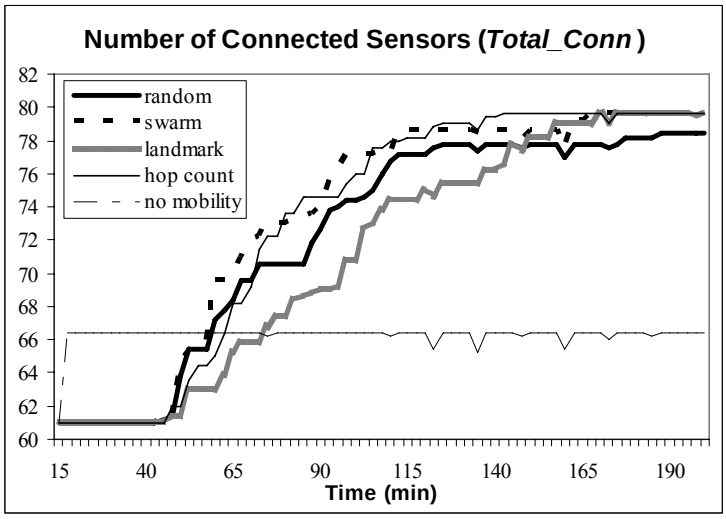

Figure 4-15 Number of Connected Sensors (*Total_Conn*)

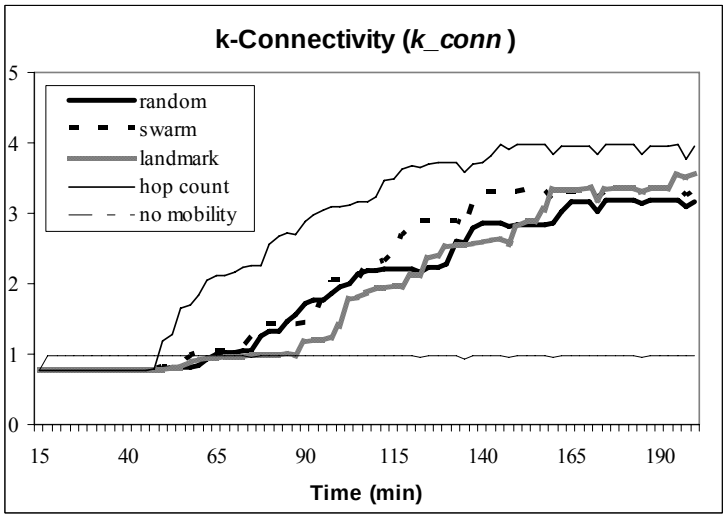

Figure 4-16 *k*-Connectivity (*k_Conn*)

From these figures, we can find that the mobility and intelligence of the robots can aid them in finding virtual targets (communication gaps). If mobile robots are not utilized in the network, the no-mobility random deployment (just throwing-in) of the new sensors can hardly improve the network connectivity.

When using robot teams, different exploration and searching algorithms result in different connectivity improvements. Potential field-based (random) exploration performs the worst. The improvement of network connectivity is slower than other intelligent mobility. Also, the final connectivity of potential field-based exploration is not as good as other mobility modes. This is because the potential field-based (random) exploration does not enable efficient cooperation among robots. Each robot makes the decision individually without any exchange of information with other robots; therefore they may search in the same regions and re-explore the covered areas.

In contrast to potential field-based (random) exploration, swarm intelligence exploration allows the robots to share their heading information so that they can be dispersed into the different regions of the environment. The simulation results show that the robots can find more targets (communication gaps) and thus improve the network connectivity within a shorter time.

Landmark-based exploration also allows the robots to be dispersed into the environment. More importantly, this exploration algorithm is able to increase the coverage of the environment through robot-landmark cooperation. However, since the information exchange between robots and landmarks may incur intense intercommunications, the communications delay is large, and thus the algorithm may be sometimes slower than other exploration or searching algorithms.

Hop-count gradient-oriented exploration performs best because it is able to lead the robots to the areas where the targets are likely to appear. Obviously, this is better than searching aimlessly.

This searching algorithm is designed for ad hoc communication networks and is shown to be effective for such distributed sensor networks. However, with respect to other kinds of targets, this algorithm may not work properly.

Videos and photos have been taken for above simulations. Please kindly refer to http://guppy.mpe.nus.edu.sg/~mpeangh/kevin to review them to have a better understanding of the system.

4.4 Experiment Tests and Discussions

In addition to simulation tests, the real robots and wireless transceivers are applied to test the efficacy of the proposed exploration and target-searching algorithms. Due to the limitation of sensing ability, our existing robots cannot differentiate or distinguish detected objects (as obstacles, robots, or embodied targets); therefore, some proposed algorithms are not implemented. In this study, the following two algorithms are tested in the real systems:

- Potential field-based exploration
- Swarm intelligence exploration

In this study, two real-robot systems are implemented, as introduced in Section 3.2.2. In both systems, the targets are embodied. They are wireless transceivers that periodically broadcast to their vicinity. When a robot is within a certain range of the target, it can receive such signals and claim it find the target.

4.4.1 Small Experiment Environment

4.4.1.1 Experiment Scenario and Settings

In this study, the small experiment environment is implemented as a simple single room application scenario. The experiment scenario is as shown in Figure 4-17:

- The environment is 6m × 5m. There are three walls inside the environment.
- There are 4 robots. They are placed in the start zone at the beginning of each run. The velocity of the robot is less than 0.8m/s in forward motion, and less than 45 degrees/s in spinning. Each robot has a wireless transceiver to send and receive information to/from other robots or the targets. The inter-robot communication range is about 1.5 to 2 meters.
- There are 2 targets deployed in the environment. They are wireless transceivers that periodically (1Hz) broadcast to neighbors. When a robot is within a certain range (0.6~1m) of a target, it can receive the broadcast of the target, and then the robot is considered to have found the target. In this case, the robot will stop moving and use its buzzer to make a "beep".

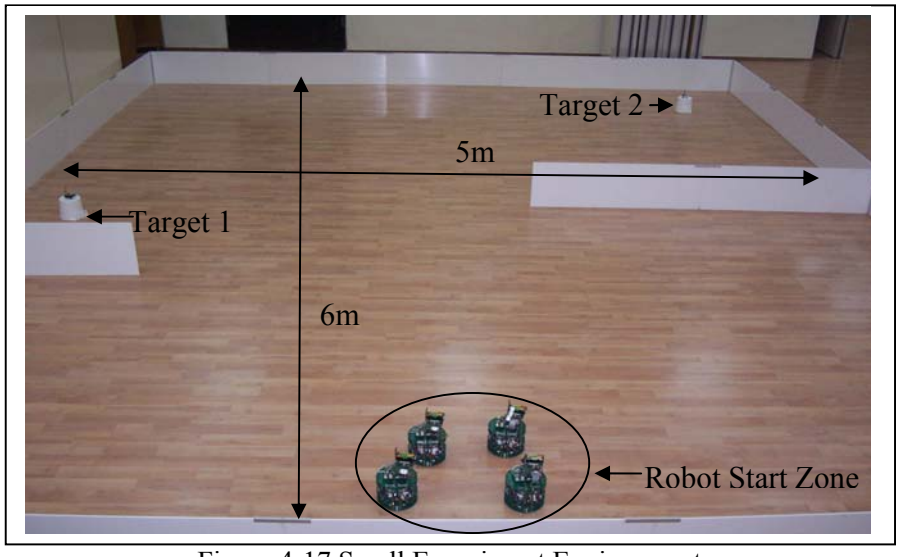

Figure 4-17 Small Experiment Environment

4.4.1.2 Experiment Results and Discussion

To compare the performance of the potential field-based exploration and swarm intelligence exploration algorithms, the following metrics are used:

- Time spent to find target 1 (*Time_T1*)
- Time spent to find target 2 (*Time_T2*)

Both *Time_T1* and *Time T2* are counted from the beginning of the experiment. Because sometimes target 1 is found first, and sometimes it is not, the *Time_T1* and *Time T2* are listed separately. It should be noted that for each run, the time to find both targets is the larger one of *Time_T1* and *Time T2*.

Because the objective of the exploration algorithms is to enable the robots to find the targets efficiently, smaller *Time_T1* and *Time_T2* correspond to better performance. For each exploration algorithm, the results are obtained from the average of 10 runs. These results are listed in Table 4-2 and Table 4-3.

Table 4-2 Small Experiment - Searching Time for Target 1 (*Time_T1*)
(seconds)

Run	Potential Field-based Exploration	Swarm Intelligence Exploration
1	15	10
2	28	7
3	48	13
4	41	65
5	13	11
6	14	10
7	63	44
8	97	7
9	89	33
10	68	20

Table 4-3 Small Experiment - Searching Time for Target 2 (*Time_T2*) (seconds)

Run	Potential Field-based Exploration	Swarm Intelligence Exploration
1	254	105
2	36	132
3	50	212
4	44	125
5	125	69
6	84	63
7	269	64
8	358	112
9	215	149
10	210	372

From above two tables, we can see that for different targets, the searching performance varies. For target 1, the swarm intelligence exploration is quite superior to potential field-based exploration (9 out of 10 runs). However, for target 2, swarm intelligence exploration is just a little better (6 out of 10 runs). Observing the experiment environment, we find that target 2 is far from the start point of robots. Before the robots reach the region close to target 2, they have already moved a long distance. During the moving, the robots have enough time to intercommunicate and therefore the swarm intelligence should be able to let them move along different directions. This will result in the separation of robots, and the separation will let robots unable to continue communicating because they become far from each other. Without intercommunication, the robots cannot use swarm intelligence. Therefore, for target 2, swarm intelligence exploration is less effective than for target 1.

In the following, the averages of *Time_T1* and the standard deviation are shown in Figure 4-18, and the averages of *Time_T2* and the standard deviation are shown in Figure 4-19. By comparing the potential field-based exploration and the swarm intelligence exploration algorithms, we find that the latter algorithm needs less time to find the target. This is consistent with the results obtained from simulations. In addition, the results show that the swarm intelligent exploration algorithm is more deterministic in that it has less deviation from the average. Comparing to the

simulation results, we may find that the deviation in experiment is large. This may be due to the imperfection of robots, such as the sensing capability. The infra-red sensors are quit sensitive to the light condition and thus the performance varies.

In these figures, we also can find that the time to find target 1 is much shorter than the time to find target 2. This might due to the following reasons:

- As shown in Figure 4-17, target 1 is about 3.5m away from the start point of robots, while target 2 is about 6m away. The robots need to travel longer distance to find target 2.
- Before reaching target 2, the robots need to avoid the wall in front of target 2. This may take some time.
- Usually target 1 is found first. When the robot find target 1, it will stay close to target 1 and send this information to the monitor center. This robot will not continue moving to search for target 2. Only the remaining 3 robots will continue searching. This means usually fewer robots are available to search for target 2.

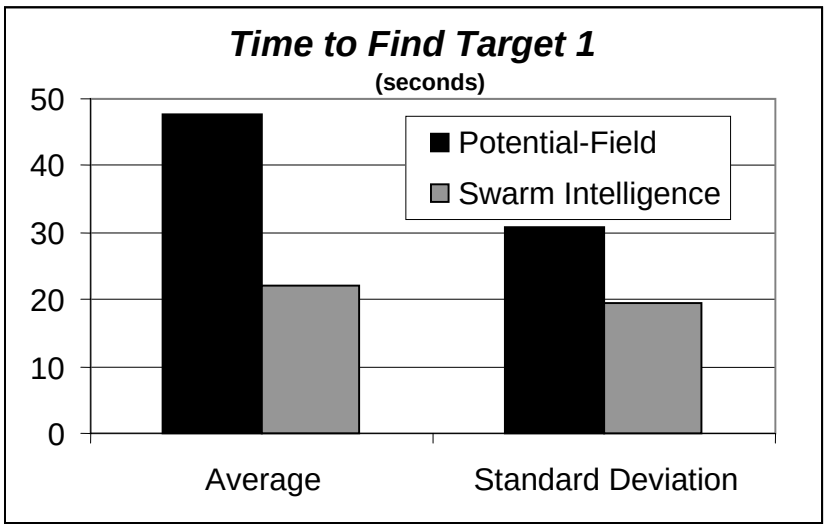

Figure 4-18 Small Experiments - Searching for Target 1 (*Time_T1*)

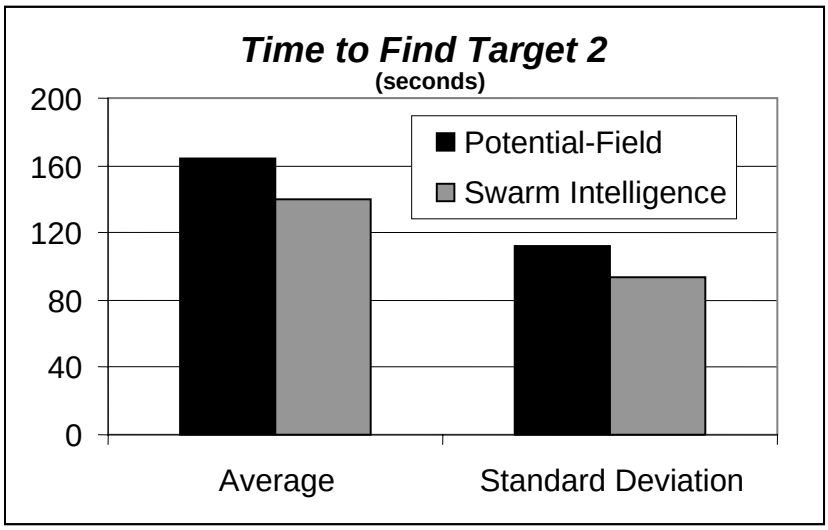

Figure 4-19 Small Experiments - Searching for Target 2 (*Time_T2*)

Figure 4-20 and Figure 4-21 may explain why the swarm intelligence exploration is superior to the potential field-based exploration. In this environment, the targets are deployed in different areas. To find them, the robots have to assign themselves to search in different areas. Using swarm intelligence, the robots are likely to choose different moving orientations, such that they may search in different areas (as shown in Figure 4-20); on the other hand, the potential field-based exploration leads the robots to search by random motions, so that the robots may not be able to spread themselves effectively in the environment (as shown in Figure 4-21). As such, the swarm intelligence exploration is more powerful in target searching.

Because the swarm intelligence exploration algorithm allows the robots to move along different orientations, the robots are more intelligent and have more deterministic (spreading) group behavior. Consequently, the searching results of the swarm intelligence exploration are more deterministic than the potential field-based exploration.

Figure 4-20 Good Cooperation
– Robots Search in Different Regions by Swarm Intelligence Exploration

Figure 4-21 Bad Cooperation
- Robots Search in the Same Region by Potential Field-based Exploration

Videos and photos have been taken for above experiments. Please kindly refer to http://guppy.mpe.nus.edu.sg/~mpeangh/kevin to review them to have a better understanding of the system.

4.4.2 Extended Experiments

4.4.2.1 Experiment Scenario and Settings

In this study, after the successful implementation in small experiment environment, an extended robot system is applied to large experiment environment to further test the feasibility and scalability of the proposed surveillance system. The extended experiment scenario is as shown in Figure 4-22:

- The environment is 15m × 10m. There are some objects (piers) inside the environment. This is to represent a large area with multiple rooms.
- There are 4 robots. They are placed in the start zone at the beginning of each run. The velocity of the robot is less than 2m/s in forward motion, and less than 90 degree/s in spinning. Each robot has a wireless transceiver to send and receive information to/from other robots or the targets. The inter-robot communication range is about 2 to 3 meters.
- There are 3 targets deployed in the environment. They are wireless transceivers that periodically broadcast to neighbors at 1Hz. When a robot is within a certain range (1.5~2m) of a target, it can receive the broadcast of the target, and then the robot is considered to have found the target. In this case, the robot will stop moving and light one of its lamp to indicate it has found a target.

Figure 4-22 Extended Experiment Environment
(The circle indicates the targets: from left to right is target 1, 2 and 3)

4.4.2.2 Experiment Results and Discussion

Similar to the small-environment experiment, the following metrics are defined and used:

- Time spent to find target 1 (*Time_T1*)
- Time spent to find target 2 (*Time_T2*)
- Time spent to find target 3 (*Time_T3*)

Time_T1, Time_T2 and *Time T3* are counted from the beginning of the experiment. The time to find all targets is the larger one of *Time_T1, Time_T2* and *Time T3*.

For each exploration algorithm, the results are obtained from the average of 15 runs. Each run is 5 minutes. The results are listed in Table 4-4 to Table 4-6. In these tables, "NA" means that the target is not found within 5 minutes.

Table 4-4 Extended Experiment - Searching Time for Target 1 (*Time_T1*) (seconds)

Run	Potential Field-based Exploration	Swarm Intelligence Exploration
1	42	46
2	101	70
3	50	63
4	58	94
5	46	70
6	48	45
7	***NA***	41
8	***NA***	52
9	51	48
10	63	40
11	50	83
12	48	55
13	45	61
14	46	146
15	41	57

Table 4-5 Extended Experiment - Searching Time for Target 2 (*Time_T2*) (seconds)

Run	Potential Field-based Exploration	Swarm Intelligence Exploration
1	90	113
2	39	47
3	38	64
4	28	48
5	125	41
6	62	40
7	27	228
8	54	102
9	45	244
10	49	58
11	57	174
12	60	42
13	61	69
14	90	83
15	243	105

Table 4-6 Extended Experiment - Searching Time for Target 3 (*Time_T3*) (seconds)

Run	Potential Field-based Exploration	Swarm Intelligence Exploration
1	***NA***	169
2	124	105
3	153	249
4	159	***NA***
5	137	260
6	***NA***	200
7	100	149
8	129	140
9	296	184
10	***NA***	200
11	***NA***	123
12	135	125
13	185	97
14	155	196
15	181	262

The above listed test results show that the potential field-based searching have less target found rate than swarm intelligence searching. The potential field-based searching's success rate is 87%, 100%, and 73%, with respect to target 1, 2, and 3; while the swarm intelligence searching's success rate is 100%, 100%, and 93%. This is possibly because the swarm intelligence searching

can better disperse the robots so that the robots can find the targets that are located at the corner, e.g., target 3.

In the following, the averages of searching time and their standard deviation is shown in Figure 4-23 to Figure 4-25. In these figures, we count the target-searching time as 360 seconds if the target is not found within the test run (300 seconds long). By comparing the potential field-based exploration and the swarm intelligence exploration, we find that for targets 1 and 3, the latter algorithm needs less time to find targets, and it is also more deterministic in that it has less deviation from the average. On the other hand, for target 2, the potential field-based searching is superior. When robots are using potential field-based searching, they are likely to move in the same direction. Therefore the potential field-based searching is better in finding target 2 because the target is in the center of the environment (as shown in Figure 4-22) and when the robots start, they are moving toward the center. However, the potential field-based searching cannot efficiently disperse the robots to different areas of the environment; therefore the time to find targets 1 and 3 is longer.

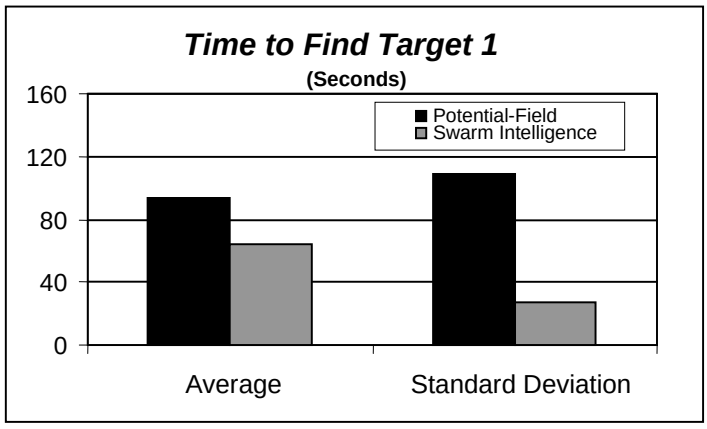

Figure 4-23 Extended Experiment - Searching Time for Target 1 (*Time_T1*)

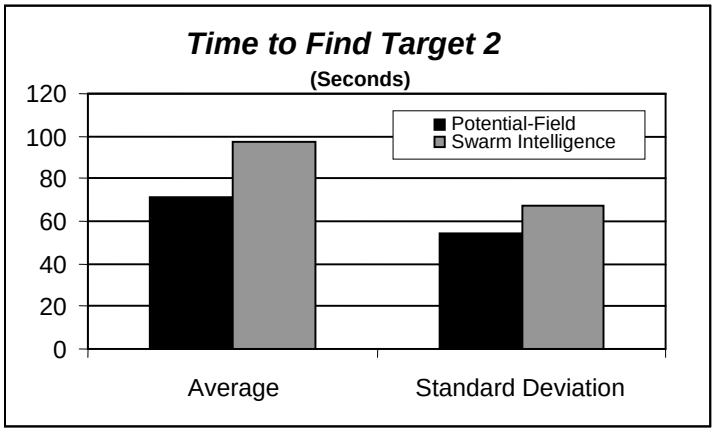

Figure 4-24 Extended Experiment - Searching Time for Target 2 (*Time_T2*)

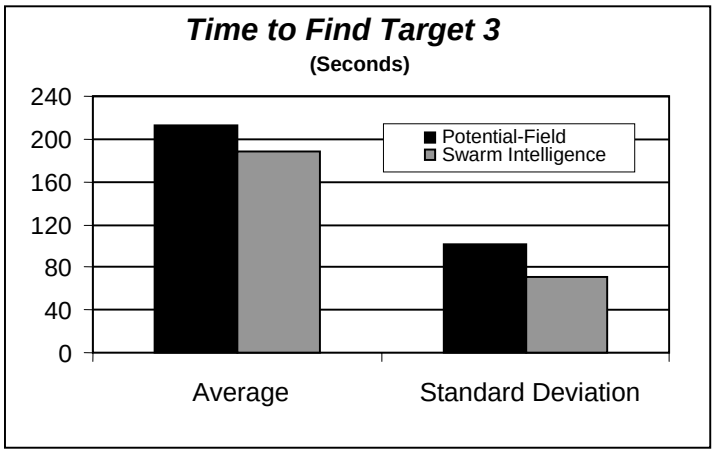

Figure 4-25 Searching Time for Target 3 (*Time_T3*)

Videos and photos have been taken for above experiments. Please kindly refer to http://guppy.mpe.nus.edu.sg/~mpeangh/kevin to review them to have a better understanding of the system.

4.5 Summary

In this chapter, the proposed exploration and target-searching algorithms are introduced (Seah et al., 2006, 2006). Without the need for map building, the robots can effectively find targets, either embodied or virtual, in unknown environments. This is enabled by the collaboration among mobile robots (potential field-based exploration, and swarm intelligence exploration), or between

mobile robots and static sensors (landmark-based exploration and hop-count gradient-oriented target searching).

The proposed exploration algorithms are simple and scalable. Potential field-based exploration algorithm and swarm intelligence exploration algorithms rely on cooperation among mobile nodes. Landmark-based exploration involves cooperation between mobile and static nodes. The paths undertaken by the mobile robots are more deterministic, and the algorithm is also able to achieve comparable performance to the exploration algorithm with map building.

The proposed hop-count gradient-oriented searching algorithm is designed specifically for searching communication gaps (virtual targets) in sensor networks. By making use of clues of the target locations, this searching algorithm is more efficient than the normal coverage-centric exploration algorithms.

The performance and efficacy of the proposed exploration and searching algorithms are demonstrated using both simulations and experiments. The simulation is implemented by the integration of robotics (Player/Stage) and communication (GloMoSim) simulators, which can reproduce the real system with the consideration of robots' system and communication system together. The experiments are implemented in both small and extended experiment environments, to test the feasibility and scalability of the system. The experiment results show that the proposed algorithms could work well in real-world applications.

5 MULTI-ROBOT TRACKING OF MULTIPLE MOVING TARGETS

In surveillance tasks, when a moving target (embodied) is found, the robots usually need to track it to achieve close and continuous monitoring. In the proposed surveillance scenario, there are multiple mobile targets and robots. Therefore, it is important to find a feasible solution to achieve cooperation among robots such that they can track as many mobile targets as possible.

In robotics research, the multi-robot tracking of multiple moving targets is also referred to as the "museum problem" or "art gallery problem". In the museum problem, the main challenge is to optimally distribute the robots such that they can observe the environment or targets perfectly. Parker (2002) proposed the Artificial Potential Field (APF)-based solution for Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT). In this approach, both obstacles and robots are mapped as repulsive force sources, and the targets are mapped as attractive force sources. Hence, the robots are able to track targets and avoid collisions at the same time. Pirjanian & Mataric (2000) introduced a more cooperative algorithm to observe moving targets. The work focuses on task allocation problems using Multiple Objective Behavior Coordination (MOBC): each task is divided into finite states, and each robot then selects its state based on its distance to the targets and the positions of other robots. Jung & Sukhatme (2001) proposed another solution for the tracking of multiple moving targets. In their work, the global distribution of robots is considered: when the robots are too densely distributed in a small sub-region, some of them will leave this sub-region and move to sparser sub-regions with fewer robots.

As compared to the above approaches, this study has more realistic assumptions and system requirements. According to the proposed surveillance scenario, the following is a short description of the environment and system:

- The environment is a large bounded 2D area.
- There are several targets moving around in the environment.
- There are several mobile robots in the environment. Each robot has a line-of-sight view within a certain range. When an object is within this range, the robot can detect the distance and angle towards this object. The robot is also able to differentiate the observed objects as obstacles, targets, or robots.
- The number, distribution and motion pattern of targets are unknown to the robots.
- The size and map of the environment are unknown to the robots. In addition, a robot is unable to localize itself in the environment.
- The summation of the sensed areas of all robots is far smaller than the size of the environment. Since the targets are mobile and the robot sensor range is limited, the robot needs to track (move together with) the targets to keep them under observation.
- Only one robot is required to track one target – there is no need to let two or more robots track the same target at the same time.
- The objective of the target tracking is to maximize the number of targets being observed (detected within the robot's sensing range) simultaneously, and minimize the member of robots that are needed to track these targets.

In this study, the surveillance system does not need to have prior knowledge of the environment or the targets to be tracked. As such, the tracking algorithms can be applied to nearly all real-world applications. However, this also increases the difficulties of the system, as the robots need to reactively find the optimal tracking strategy for the environment and the targets.

The robots in this surveillance system do not have (accurate) location information because they are not equipped with expensive sensors and devices (e.g., Global Positioning System is

expensive and not applicable in indoor environments). In addition, the robots should use no or minimal direct intercommunications due to the constraints of transceivers (cost, bandwidth and power). These limitations impose more difficulties in achieving optimal control, but ensure the applicability of the systems in real-life applications.

To solve the above-mentioned problems, this thesis develops an advanced potential field-based tracking algorithm to achieve multi-robot cooperative tracking (Liu et al., 2003, 2004a, 2004b). Two reinforcement learning-based learning algorithms are also developed to improve the adaptation of robots to the environment and targets (Liu et al., 2004c, 2005a, 2005b, 2006). In the following parts of this chapter, the advanced potential field-based tracking algorithm is introduced in Section 5.1 and the two learning algorithms are presented in Section 5.2. Following this, Section 5.3 presents the implementations of these proposed algorithms in simulations. Corresponding results and discussions are also presented in this section. Finally, Section 5.4 concludes this chapter.

5.1 Cooperative Artificial Potential Field-based Tracking

In multi-robot tracking of multiple moving targets, the main challenge is to optimally assign the targets to the robots, such that the robot resource can be fully utilized to track as many targets as possible. This is a target selection problem. As shown in Figure 5-1, a good target-selection algorithm should allow robots to choose the most suitable targets to track (the left four figures). On the other hand, it is not desirable to waste the robot resource to track the same target, or lose some targets (the right two figures).

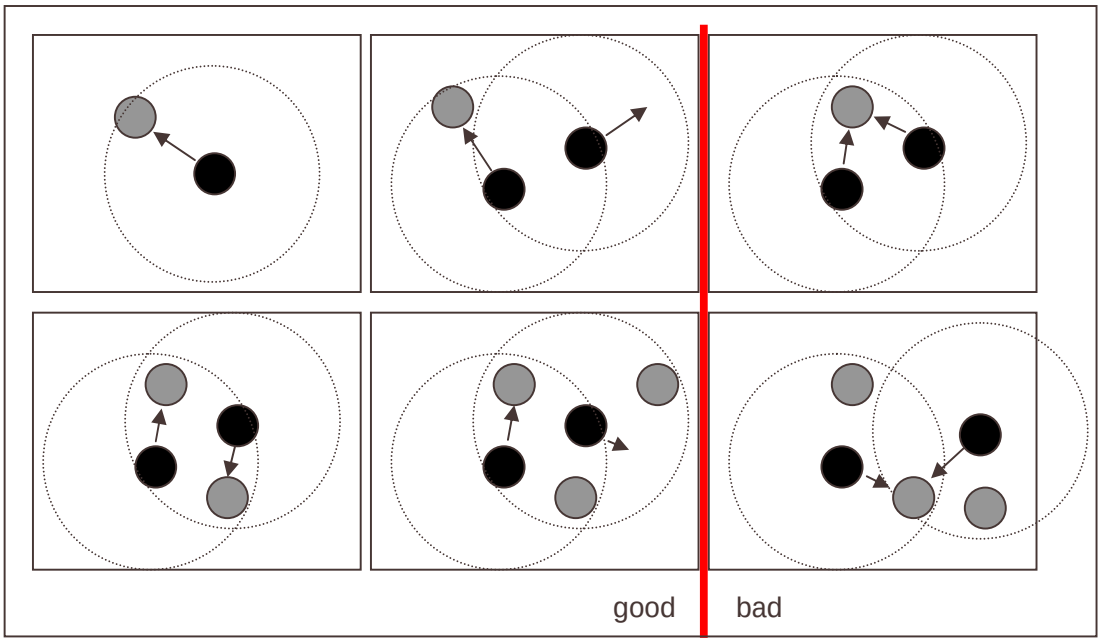

Figure 5-1 Target Selection
(black dot: robot; grey dot: target;
dotted circle: sensing range of robot; arrow: moving direction of robot)

5.1.1 Pure APF-based Control and All-Adjust Heuristic

According to the constraints of the system, the robots have limited sensing range and no location information, and should use no or minimal direct intercommunications. This increases the difficulties in achieving the desired cooperative target selection. To solve the target-selection problem under such hard limits, Artificial Potential Field (APF)-based control is proposed and applied by many researchers (Parker, 2002). The implementation of the APF controller for tracking is similar to the APF controller for exploration (Section 4.1.1). The main difference is in the calculation of attractive forces. In the APF exploration controller, the attractive force is randomly generated, but in the APF tracking controller, the attractive force is toward the target(s).

Pure APF control for target tracking sums the virtual attractive and repulsive forces to drive the robot, as shown in Equation (5.1):

$$\vec{F}_{R_i} = \sum_{\substack{T_j \in \text{detected} \\ \text{targets}}} \vec{A}_{R_i,T_j} + \sum_{\substack{O_l \in \text{detected} \\ \text{objects}}} \vec{R}_{R_i,O_l} \tag{5.1}$$

In this equation, $\vec{F}_{R_i}$ is the summation of the attractive and repulsive forces for robot R_i; $\vec{A}_{R_i,T_j}$ is the virtual attractive force from robot R_i to target T_j. The orientation of the force is the angle to T_j, and the magnitude is a fixed value, e.g., 1.0, or a value that is dependent on the distance between R_i and T_j; $\vec{R}_{R_i,O_l}$ is the virtual repulsive force from detected object O_l to robot R_i. The orientation of the force is the angle to R_i, and the magnitude is a fixed value or a value that is dependent on the distance between R_i and O_l. Neighboring targets that are within the sensor range of R_i are "detected targets"; neighboring robots and obstacles that are within the sensor range of R_i are "detected objects".

APF control is simple and scalable. It can achieve cooperation among robots without any need for intercommunications. However, pure potential field-based control may be unable to achieve the desired level of cooperation in most cases. For example, if two robots detect the same target, both of them will track this target and form a triangular pattern. This is not the optimal level of cooperation: the robot resource is underutilized because one of the robots should leave to search for other targets to maximize the number of targets being observed. This situation is depicted in Figure 5-2. In this figure, the two robots find the same target and are also able to detect each other. The summation of the virtual attractive and repulsive forces allows the robots to form a triangular pattern to track the same target.

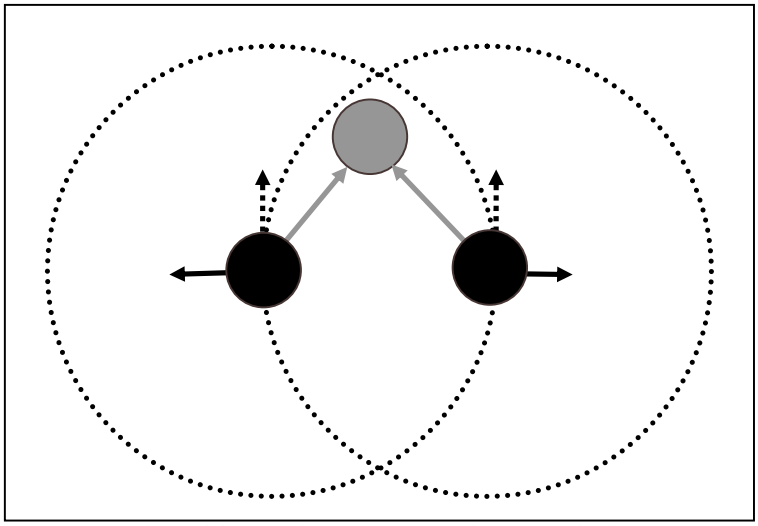

Figure 5-2 Undesirable Tracking – Two Robots Track the Same Target
(Gray dot: target; black dot: robot; dotted circle: sensor range; gray arrow: attractive force; black arrow: repulsive force; dashed arrow: summation of forces)

To avoid the "triangular pattern" of pure APF, a weight $w_{R_i}^{T_j}$ can be assigned to the attractive force for each robot as shown in Equation (5.2):

$$\vec{F}_{R_i} = \sum_{\substack{T_j \in \text{detected} \\ \text{targets}}} w_{R_i}^{T_j} \vec{A}_{R_i,T_j} + \sum_{\substack{O_l \in \text{detected} \\ \text{objects}}} \vec{R}_{R_i,O_l} \tag{5.2}$$

In this equation, $w_{R_i}^{T_j}$ is the weight assigned to change the attractive force from robot R_i to target T_j; other variables are defined in the same manner as in Equation (5.1).

By examining Equation (5.2), we can see that if the weight of the attractive force is zero, the robot will only have one behavior, i.e., avoid neighboring robots or obstacles; if the weight of attractive force tends towards infinity, the robot will also have a unique behavior, i.e., track targets. Changing the value of the weight means changing the relative preference between two behaviors "tracking target" and "leaving neighboring robots/obstacles".

An algorithm is proposed by Parker (2002) to adjust this weight dynamically by letting one robot decrease the weight when it finds that another robot is also tracking the same target (Algorithm 5.1).

Algorithm 5.1 All-adjust heuristic of pure potential field-based control (for Robot R_i)

Step 1. Set initial force to move as $\vec{F}$. The orientation of $\vec{F}$ is a random value that is uniformly distributed between [0, 2π); the magnitude of $\vec{F}$ is a fixed value, e.g., 1.0.

Step 2. Scan the surrounding environment; suppose that n_T targets and n_O objects (obstacles and other robots) are found within the sensor range. "*detected targets*" = $\{T_1, T_2, \ldots, T_{nT}\}$; "*detected objects*" = $\{O_1, O_2, \ldots, O_{nO}\}$.

Step 3. If $n_T > 0$, let $\vec{F} = 0$, and

for $j = 1$ to n_T

If a robot is found around target T_j (e.g., robot R_k is found and its distance to target T_j is less than the sensor range of R_k), let $\vec{F} = \vec{F} + w_j \vec{A}_{R_i,T_j}$. The value of w_j is the All-Weight Decrease Ratio (AWDR between [0, 1]).

Else, let $\vec{F} = \vec{F} + \vec{A}_{R_i,T_j}$

Step 4. If $n_O > 0$, let $\vec{F} = \vec{F} + \sum_{l=1}^{n_O} \vec{R}_{R_i,O_l}$.

Step 5. Let robot R_i move under the virtual force $\vec{F}$.

Step 6. Goto Step 1.

The drawback of the all-adjust heuristic of potential field-based control is that all robots who find the same target will reduce their attractive forces toward this target. If the reduction is small, all these robots may continue tracking the target (as shown in the left of Figure 5-3); if the reduction is large, all these robots may leave the target (right of Figure 5-3).

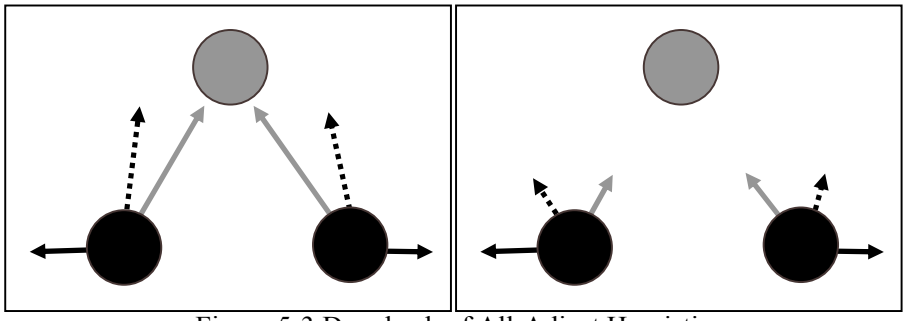

Figure 5-3 Drawback of All-Adjust Heuristic
(Gray dot: target; black dot: robot; gray arrow: attractive force; black arrow: repulsive force; dashed arrow: summation of forces)

5.1.2 Selective-Adjust Heuristic of Pure APF-based Control

To solve the problem associated with the pure APF-based control and the all-adjust heuristic, a selective-adjust heuristic is proposed in this study (Liu et al., 2003, 2004a, 2004b). This algorithm is shown in Algorithm 5.2.

Algorithm 5.2 Selective-adjust heuristic of potential field-based control (for Robot R_i)

Step 1. Set initial force to move as $\vec{F}$. The orientation of $\vec{F}$ is a random value that is uniformly distributed between [0, 2π); the magnitude of $\vec{F}$ is a fixed value, e.g., 1.0.

Step 2. Scan the surrounding environment; suppose that n_T targets and n_O objects (obstacles and other robots) are found within the sensor range. "*detected targets*" = $\{T_1, T_2, \ldots, T_{nT}\}$; "*detected objects*" = $\{O_1, O_2, \ldots, O_{nO}\}$.

Step 3. If $n_T > 0$, let $\vec{F} = 0$, and

for $j = 1$ to n_T

If one or more robots are found near target T_j and robot R_i is NOT the nearest robot to target T_j, let $\vec{F} = \vec{F} + w_j \vec{A}_{R_i,T_j}$. The value of w_j is the Selective-Weight Decrease Ratio (SWDR) between [0, 1].

Else, let $\vec{F} = \vec{F} + \vec{A}_{R_i,T_j}$

Step 4. If $n_O > 0$, let $\vec{F} = \vec{F} + \sum_{l=1}^{n_O} \vec{R}_{R_i,O_l}$.

Step 5. Let robot R_i move under the virtual force $\vec{F}$.

Step 6. Goto Step 1.

Comparing all-adjust and selective-adjust heuristics, we may find that the former is simpler but may not achieve the best cooperation because all robots may leave or continue to track the same target. The latter heuristic is more effective because it considers the distance between robots and the target; therefore only the most feasible (nearest) robot will continue tracking the target. This is a high-level cooperation algorithm for multiple target tracking.

5.1.3 Summary

In this subsection, the traditional potential field-based control and the all-adjust heuristic are reviewed. To solve the problems associated with these algorithms, the selective-adjust heuristic is proposed in this thesis. This heuristic can achieve cooperation among robots by letting them choose suitable targets to track. Furthermore, this algorithm is simple and scalable for a large number of robots and targets as it does not require any explicit intercommunications. Without the need for global information, e.g., map and location information, each robot makes its decisions based individually on local sensing.

While the APF-based control and its heuristics are simple and efficient, it is difficult to find the optimal parameter value for the controller, e.g., the weights, especially in complex and dynamic application environments. To solve this problem, intuitively, machine learning algorithms can be

applied to allow the robots to learn and adapt to the environment to achieve optimal cooperation for the tracking of multiple moving targets. This motivates the study on machine learning approaches, the details of which will be introduced in the next subsection of this chapter.

5.2 Learning of Cooperative Tracking

5.2.1 Traditional Reinforcement Learning and Its Constraints

As introduced in Section 5.1, multi-robot tracking of multiple moving targets is a target-assignment problem that aims to assign the target to the most suitable and feasible robots. In the proposed application scenario, the number, location, and mobility pattern of targets are unknown. This increases the uncertainty for system design, because important parameters, e.g., the weight decrease ratio, cannot be pre-determined for the hardcoded controllers to achieve the desired cooperative target assignment.

In robotics research, the desired target assignment for multi-robot tracking requires task-level cooperation, in which the mission is broken down into tasks, and robots choose different tasks (roles) according to the current state. Each robot will also behave differently, depending on its assigned task. However, to achieve mission decomposition, task allocation and conflict resolution, the designer needs to predict all possible scenarios and preset corresponding actions for each robot. Such development and coding work is undesirable and can be extremely difficult, especially when the mission is complex and the robots are heterogeneous. Therefore, it is desirable to let mobile robots learn to work cooperatively through interactions among robots and the feedback from the environment. This can generate appropriate robot behaviors without the need for human design or pre-programming of behaviors.

The basic concept of reinforcement learning is to find the optimal control policy that chooses the appropriate action under any given state; in other words, it should be able to find the optimal link/mapping from states to actions. Reinforcement learning is a simple but powerful learning algorithm that is model-free, does not require strict supervision, and can achieve optimal subject to user-defined criteria (Sutton & Barto, 1998). In addition, reinforcement learning provides a natural fit for behavior-based control, which requires the robot to "select" optimal actions under any given state (Mataric, 2001). For example, a robot can use the reinforcement learning algorithm to learn the elementary behavior "avoid obstacles", such that when it is "near to an obstacle" (high-level state), it will carry out the (high-level) action "makes a detour along the boundary of the obstacle".

In the last two decades, reinforcement learning has been extensively studied for multi-robot concurrent learning of cooperative behaviors. However, to apply reinforcement learning to behavior-based control, the designers usually need to discretize the continuous input state space and output action space (Chu & Hong, 2000). The problem associated with discretization is that if the discretization is too coarse, some states may be hidden and the optimal control policy can not be found; if the discretization is too fine, the states cannot be generalized and the huge state/action space will negatively impact the learning speed. In addition, if the states and actions are discretized and finite, the behaviors will also be discrete and finite because the robot can only perform one action corresponding to a single behavior at any one time. This contradicts the human reasoning that the optimal solution to accomplish a task may comprise the concurrent execution of several elementary behaviors. Furthermore, switching between discrete behaviors usually results in unsmooth control, which is undesirable in most scenarios.

With respect to this problem, several methods have been proposed to enable reinforcement learning in continuous space without discretization. The function approximation approach (Boyan

& Moore, 1995) and HEDGER (Smart & Kaelbling, 2000) apply a generalizing function approximator to estimate the state-action value instead of using discrete lookup tables. Doya (1996) and Hagen (2001) proposed reinforcement learning to derive optimal feedback control laws for linear/nonlinear systems. However, these approaches usually assume that the environment model is known and may incur a heavy computational burden if the training data set is large.

Another class of solutions is to integrate reinforcement learning with Fuzzy Inference Systems (FIS) by allowing the reinforcement learning module to learn/tune the fuzzy rules for the FIS. The FIS can then retrieve continuous and infinite states and perform the corresponding actions. Jouffe (1998) proposed dynamic programming algorithms that are applied in a four-layer FIS scheme for online tuning of the number and positions of the input fuzzy labels (weights). Yan et al. (2001) introduced a reinforcement learning algorithm for learning fuzzy rules of a Takagi-Sugeno-type FIS. Ye et al. (2003) applied reinforcement learning methods to maintain the correctness, consistency and completeness of fuzzy rules. These deliberatively designed approaches can tune the fuzzy inference systems to achieve satisfying performance; however, the control architecture and learning algorithm are usually complex and the applications are mostly for low-level control involving simple tasks and missions, e.g., approaching targets with obstacle avoidance.

Another drawback of multi-robot concurrent learning is that the traditional single-agent/robot reinforcement learning may not work appropriately in multi-robot domains. Two basic assumptions, Markov decision process and stationary environment, which are usually valid in the single-robot domain, are inapplicable in multi-robot domains due to the interactions among the concurrently learning robots (Kaelbling et al., 1996). This is also known as the convergence or stability problem of multi-robot (agent) learning. One class of solutions to address this problem is to estimate the influence of other robots and consider the process as semi-Markovian and pseudo-

stationary for an individual learning robot (Kawakami et al., 1999). Another class of solutions is to coordinate or schedule the distributed learning processes to reduce the interference (Uchibe et al., 1998; Asada et al., 1999; Ikenoue et al., 2002). However, the coordination and scheduling of the learning processes have to be deliberatively designed and usually require explicit communications among learning robots.

To address the above problems with the finite and discrete state/action spaces, this study proposes two reinforcement learning-based controllers: (i) reinforcement learning in behavior-based control networks; and (ii) fuzzy reinforcement learning. In addition, a distributed learning coordination algorithm is developed to reduce or eliminate the interference among concurrently learning robots.

5.2.2 Reinforcement Learning in Behavior-based Control Networks

5.2.2.1 Proposed Learning Controller

To apply reinforcement learning in multi-robot systems, there are two problems to be addressed:

- How can the optimal combination of elementary behaviors for cooperation be generated based on low-level input states and output actions?
- How can concurrent learning processes be coordinated efficiently in a distributed manner?

To overcome the drawbacks of a discrete and finite number of elementary behaviors, the integration of reinforcement learning with behavior-based control networks is proposed (Liu et al., 2004c, 2005b, 2006). The architecture of this learning controller is shown in Figure 5-4 (inside

the dotted rectangle). It includes a behavior-based control network module and a reinforcement learning module.

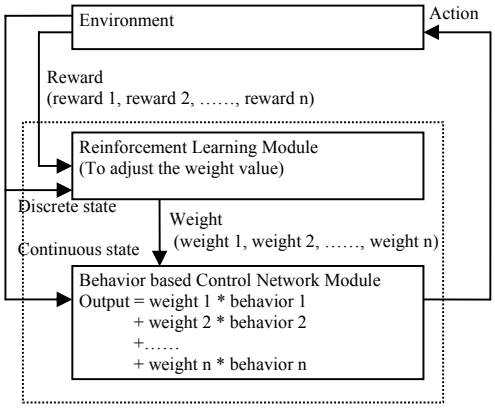

Figure 5-4 Reinforcement Learning in Behavior-based Control Network

The behavior-based control network is created according to the human knowledge of the robot, environment and mission. In this network, the elementary behaviors are represented by control rules and equations. For example, the elementary behavior "obstacle avoidance" may be represented by an equation; and "target tracking" may be a batch of fuzzy rules with corresponding membership functions. Each elementary behavior can retrieve a continuous and infinite number of input signals, and generate a continuous and infinite number of output commands. Finally, the overall output is the summation of weighted outputs of all elementary behaviors. In this behavior-based control network, the weight is the key to combining different elementary behaviors: if the weight assigned to a particular behavior is large, the robot is more likely to perform this behavior; otherwise the robot is reluctant to perform this behavior.

In the proposed learning controller, the reinforcement learning module is integrated with the behavior control networks. The aim of this learning module is to adjust the weight inside the control network, thus affecting the combination of elementary behaviors. This is the key to generating an optimal combination of behaviors. By retrieving states and rewards, the learning

module can gradually find the appropriate weight value to be assigned to each elementary behavior; therefore, the optimal control policy is learned. It should be noted that the reinforcement learning module needs to retrieve rewards corresponding to each behavior; otherwise, the learning module will be unable to estimate the performance or results of taking the behavior. For example, if the performance of the behavior "avoid obstacles" is unsatisfactory, a negative reward (penalty) should be given to indicate that the weight of "avoid obstacles" has to be adjusted.

In general, the proposed learning controller has the following properties:

- The behavior-based control network is designed based on human experience. The control rules or equations are the representations of elementary behaviors.
- In the control network, the overall output behavior is the summation of weighted elementary behaviors. The output control command is smooth and continuous.
- The aim of the reinforcement learning module is to adjust the weight in the control network to achieve an optimal combination of elementary behaviors. In other words, the output is not the "selection" of exclusive discrete behaviors, but a method for "combining" them to generate a continuous and infinite number of behaviors.
- The reinforcement learning module still works within discrete and finite space. However, in the macro view, the learning controller works in continuous space as it can retrieve low-level inputs and generate appropriate low-level outputs representing a continuous and infinite number of possible behaviors.

To implement the proposed learning controller for the tracking problem, the key is to learn the optimal weights for potential field-based control so that the robots can track targets cooperatively and efficiently.

Figure 5-5 shows the flowchart of the learning process in the learning module of the learning controller. The main research issues include state/action definition, reward generation, state-action value update and action selection. The distributed learning coordination algorithm is also important in the learning process.

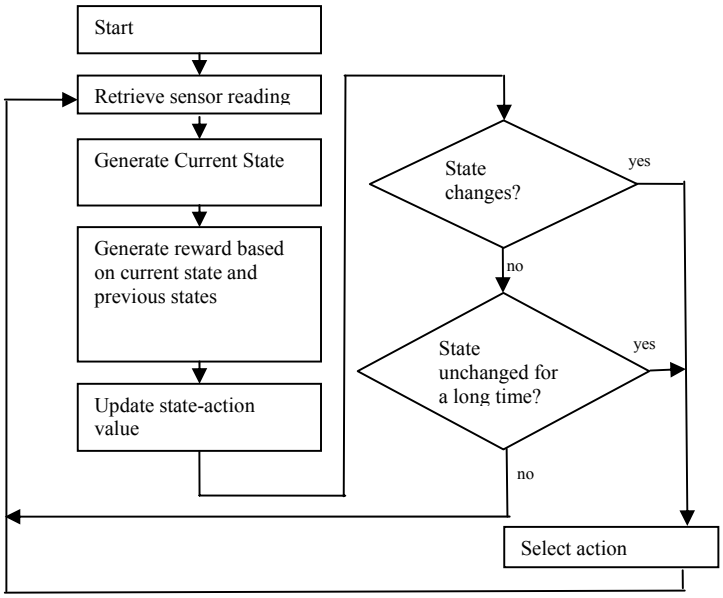

Figure 5-5 Flow Chart of Learning Process

In the following, the important learning issues are introduced.

5.2.2.2 State Definition and Reward Generation

In the museum problem, each robot may encounter many situations. To make the learning simple, yet without losing generality, this study defines the input state as the number of targets and robots detected. For example, if two targets and one neighboring robot are detected, the state is (*2, 1*). The output of the learning module (action) is the weight of the attractive force.

When the states are detected, it is important to generate appropriate rewards based on these states. This is because the reward represents the objective of the system designer and can directly affect the learning results. Since task-level cooperation is desired, the following behaviors should be encouraged: (i) track target; and (ii) leave the target being tracked by other robots. For this purpose, this study defines four types of rewards:

- *Reward_TT*: track-target reward (positive) – awarded if the robot tracks targets.
- *Reward_NR*: near-robot penalty (negative reward) – awarded if the robot detects the presence of other robots.
- *Reward_SC*: state-change reward (positive or negative) – if there are fewer neighboring robots and more targets in the new state, the reward is positive; otherwise, the reward is negative.
- *Reward_WT*: waste time penalty (negative reward) – awarded if the robot tracks a target that is simultaneously being tracked by other robots.

For each individual robot, the above mentioned rewards are generated by its local sensing. For example, in case robots *A* and *B* are tracking the same target, from the perspective of robot *A*, it will be awarded with *Reward_TT,* and *Reward_NR* and *Reward_WT* if *B* is detected.

5.2.2.3 State-Action Value Update

In reinforcement learning, the essential problem is to find the appropriate state-action value to represent the effectiveness (expected reward) of taking the action in the state. The learning process has to update the state(*s*)–action(*a*) value, *Q(s, a)*, based on the reward received. In this approach, this value is updated by the Q-function (5.3) as introduced in Algorithm 5.3.

Algorithm 5.3 Q-Learning Algorithm

Step 1. Initialization: define the state pool $\{s_i\}$ to represent the possible states of the environment; define the action pool $\{a_j\}$ to represent the possible actions to perform; set current time step $t = 0$; set the initial state-action values $\{ Q(s_i, a_j) \}$. Here, $Q(s_i, a_j)$ is the value representing the expected reward for performing action a_j under state s_i.

Step 2. Detect the state of the environment and choose s_t to represent the real state.

Step 3. Choose an action a_t, such that $a_t = arg\ max\ (Q(s_t, a) + RandomFactor)$.

Here, *RandomFactor* is a random value used to avoid local optimality of the selection of actions.

Step 4. Perform action a_t, then detect the new state s_{t+1} and receive the immediate reward r_t.

Step 5. Update the Q value by the following Q-function (5.3), in which s_t, a_t, r_t, α, γ, s_{t+1} and a_{t+1} represent state, action, reward, learning rate, discount rate, next state and next action respectively (Sutton and Barto, 1998):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{5.3}$$

Step 6. Check whether the termination condition is satisfied,

If yes, then stop;

Else, let $t = t + 1$, go to Step 3.

5.2.2.4 Action Selection

Each time the state changes, the robot will reselect the action (weight). In addition, if the state is unchanged for a long period of time (N simulation steps), the robot also reselects its action (weight) to accelerate the learning speed.

In action selection, it is important to allow the robots to explore all possibilities (actions) with respect to each single state. In this way, the robot can then find out the most suitable action for any particular state. On the other hand, the robot also needs to exploit the learning progress to execute the "good" actions that have already been learnt. This helps to reinforce the appropriate behaviors. As such, when selecting an action, the robot should both explore and exploit the action space: an exploration factor (random value) is added to the real state-action value and the action with the highest resultant value will be chosen. It should be noted that this exploration factor affects the action selection only; it has no effect on the state-action value.

5.2.2.5 Learning Coordination

In addition to the problems associated with discrete behaviors, another research issue is the coordination of concurrent learning processes. In this study, the proposed distributed learning coordination algorithm is integrated with the state-action update step in the learning controller (shown in Figure 5-5).

In multi-robot concurrent learning, the interferences among robots may invalidate the assumptions of the Markov Decision Process and stationary environments. To address this problem, a solution inspired by natural human behavior is proposed. Assume that there are two people approaching each other along the corridor and they want to avoid collision. Both of them may have to "struggle" through several rounds before they can find the best solution. In real life, usually one person (say *A*) will fix his/her policy first, e.g., keeping left; then the other person (say *B*) can choose the opposite side. In this encounter case, the optimal cooperation is that the two people choose opposite sides. Whatever *A* chooses initially, if *B* can finally learn to choose the opposite side, the resultant control policy is optimal. Many real-world applications have the

same property: even if the learning process of one robot stops at a very early stage, the resultant control policy of the whole system can still be optimal because the other learning robots can eventually find the appropriate control policy to respond to the previous one. In other words, in cooperative multi-robot systems, the "optimization" lies in the relationship among robots.

The distributed learning coordination algorithm is proposed based on the above considerations. As shown in Algorithm 5.4, for any particular robot, if the value of the best action is much larger than other actions in any one state, it will stop learning in this state and thereafter always choose this best action when it is in this state. In other words, a robot will fix its control policy when it feels that it has learned enough; the future improvement of the group performance is then left to other learning robots. This learning coordination algorithm is embedded inside the state-action update step in the proposed learning process (Figure 5-5). Before a robot updates its state-action link value, it will run the local learning coordination algorithm to decide if it should change the state-action value. This learning control algorithm is entirely distributed and does not require any intercommunication among robots. The algorithm is effective for the corridor encounter applications described previously.

Algorithm 5.4 Learning Coordination

Step 1. For each state s_i in the state space, set flag(s_i) = 0

Step 2. Check the state-action value for each state s_i: $Q(s_i, a_j)$, where a_j is the jth possible action.

If there exists a "a_j", such that $Q(s_i, a_j) > \frac{1}{m}\sum_{n=1}^{m} Q(s_i, a_n) + Threshold$, then set flag($s_i$)=1. Here, m is the number of possible actions; and *Threshold* is a positive value which indicates the minimum difference between the value of the best action and the average value of all actions before the robot can stop learning.

Step 3. If flag(s_i) = 0, $Q(s_i, a_j)$ will be updated, otherwise $Q(s_i, a_j)$ will not be updated.

5.2.2.6 Summary

In this subsection, a reinforcement learning controller is introduced. In this controller, reinforcement learning is integrated with the behavior-based control network. This controller aims to find the optimal combination of different elementary behaviors to achieve optimal control. In addition, the controller is totally distributed and relies solely on local sensing. This ensures the scalability of the robot system. While no intercommunication is required, a distributed learning coordination algorithm is developed to help eliminate the interference among concurrently learning robots.

5.2.3 Fuzzy Reinforcement Learning

5.2.3.1 Integrated Fuzzy Reinforcement Learning Controller

The proposed reinforcement controller (in Section 5.2.2) can achieve continuous control of the robots; however, the discrete state and action spaces have to be defined for the algorithm to work properly. It is not easy to find the appropriate thresholds to perform discretization in most real applications. Therefore, a non-discretization-based solution for reinforcement learning is required.

To address the above mentioned problem, a learning controller integrating reinforcement learning and fuzzy logic is proposed in this thesis (Liu et al., 2005a). This controller is designed with the following objectives:

- Enable robots to learn through their input/output behaviors.
- Enable robots to learn a cooperative control policy by distributed (local) learning processes.

- Coordinate the distributed learning processes to avoid undesired learning results (local suboptimal control policy or cyclic switching of control policies)

The controller is depicted in Figure 5-6. The blocks in the structure represent three main research issues: (i) Fuzzy Inference System (fuzzifier, fuzzy inference system, and defuzzifier); (ii) Reinforcement Learning of Fuzzy Rules (reward generator and reinforcement learning module); and (iii) Coordination of Concurrent Learning Processes (reinforcement learning module). In the following parts of this subsection, these modules are introduced.

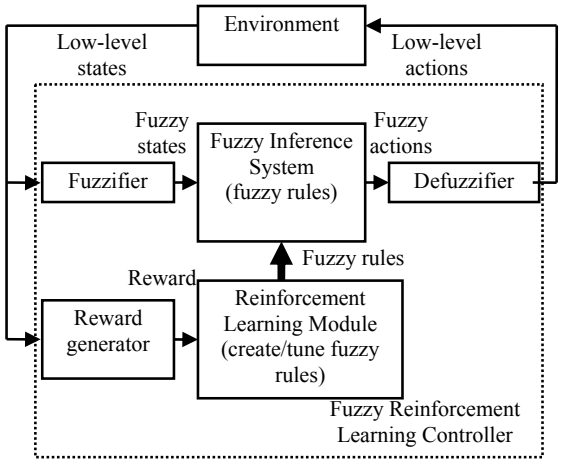

Figure 5-6 Fuzzy Reinforcement Learning Controller

5.2.3.2 Fuzzy Inference System

Fuzzy inference system is based on the concept of fuzzy sets. In this approach, the low-level environment information is fuzzified to fuzzy states, and the low-level actuator commands are fuzzified to fuzzy actions. The fuzzification methodology has the following properties:

- With respect to the environment input, each classification of information (or sensor reading) is represented by one fuzzy state. This fuzzy state (action) covers the whole data range.
- With respect to the actuator output, each classification of commands is represented by one fuzzy action. This fuzzy action comprises a group of commands that have common properties.
- The design of fuzzy states and actions is based on prior knowledge of the mission, robot and environment.

Compared to typical fuzzy inference systems, the definition of the fuzzy states in this approach is "fuzzier". This is due to the following reasons:

- In this approach, one fuzzy state represents the whole data range of one classification of environment information. It does not describe the "fuzzy value" of the environment variable (e.g., sensor input); instead, it provides membership values for the entire range of the environment variables. For example, in Faria & Remero's approach (2000), there are four fuzzy states, namely "Nearest", "Near", "Far" and "Farthest", which are used to describe the distance to the target (Figure 5-7-a). However, in the approach presented in this thesis, there is only one fuzzy state "target is found" to represent the information of distance to the target (Figure 5-7-b). This fuzzy state covers the whole data range of distance, and its membership degree (value) is given based on human knowledge: if the target is near, the target may have greater influence on the robot; therefore, the value of "target is found" should be higher.
- In this fuzzy reference system, it is possible that several fuzzy states are activated concurrently at the same time. This enables the robot to learn several basic behaviors for each fuzzy state concurrently.

- By this fuzzification methodology, fewer fuzzy states are defined and thus used in learning. This may help avoid the curse of dimensionality in reinforcement learning (Kaelbling et al., 1996).

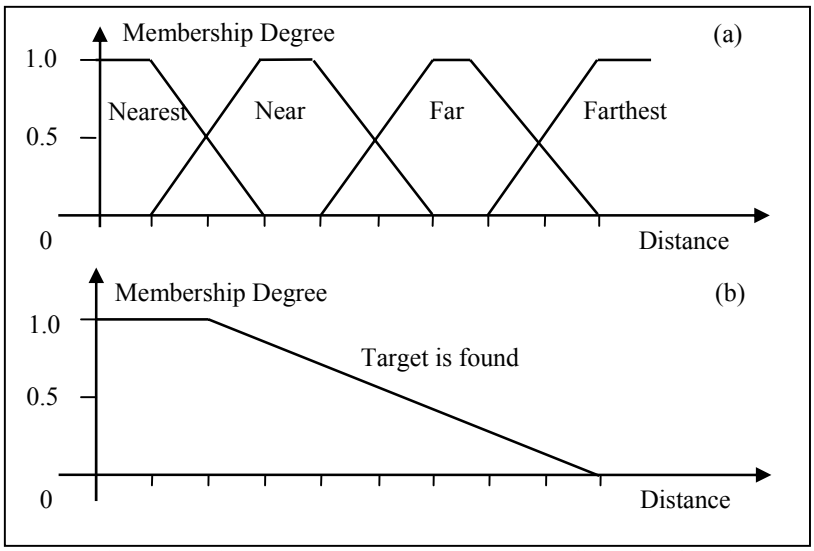

Figure 5-7 Different Definition of Fuzzy State - (a) Traditional; (b) This Approach

In the proposed fuzzy inference system, the definition of fuzzy actions is also based on human knowledge. One fuzzy action is defined to represent each type of behavior. The membership function demonstrates the relationship between the behavior level (strength) and the environment status. For example, the membership function for the fuzzy action "track target" may be defined as follows: when the target is near (distance is small), the robot does not need to put much emphasis on tracking it; therefore, the level (strength) of the behavior "track target", i.e., the membership value, is low.

When the fuzzy states and actions are defined, the fuzzy inference system can make decisions based on fuzzy rules. In this approach, the format of the fuzzy rules is defined as rule $r_{s,a}$: "IF s, THEN a". In this rule, $r_{s,a}$, s and a represent fuzzy rule, fuzzy state and fuzzy action respectively.

In the proposed fuzzy inference system, the total number of fuzzy rules to be tested is equal to *m* times *n*; where *m* is the number of fuzzy states and *n* is the number of fuzzy actions. The aim of the reinforcement learning module is to find the optimal fuzzy rules with respect to each input fuzzy state; the results are *m* fuzzy rules. For example, if the fuzzy inference system has four fuzzy states and three possible fuzzy actions, then the aim of reinforcement learning is to find four optimal fuzzy rules for each of the fuzzy states.

During learning, the robot will select fuzzy actions with respect to current fuzzy states according to the fuzzy rules being learned (or already learned). In this approach, more than one fuzzy state may happen concurrently, so that several corresponding fuzzy actions may be activated together. In this case, the output of the fuzzy inference system, *FIS_output*, is the summation of the fuzzy actions activated by fuzzy states, as shown in Equation (5.4).

$$FIS\_output = \sum_{\text{activated state } i} (mvfs \cdot mvfa)_i \tag{5.4}$$

In this equation, *mvfs* is the membership value of activated fuzzy state (as shown in Figure 5-7); *mvfa* is the membership values of corresponding fuzzy action. *mvfa* represents the level of fuzzy action according to current condition. For example, the fuzzy action "track target" may have different membership value according to the distance to target. If the target is close, *mvfa* of "track target" is small; if the target is far, *mfva* is large. As a result, *mvfs* multiply *mvfa* indicates the level of the fuzzy action to be performed by the robot under given fuzzy state. It should be noted that *mvfs* and *mvfa* are different for each activated fuzzy state.

5.2.3.3 Reinforcement Learning of Fuzzy Rules

The reinforcement learning issues for the proposed fuzzy reinforcement learning controller are similar to the reinforcement learning controller introduced in Section 5.2.2. Therefore, only the differences between these two controllers are introduced in the following.

The fuzzy inference system makes decisions based on fuzzy rules. While the fuzzy rules in most fuzzy inference systems are deliberatively designed by the designer, the proposed fuzzy reinforcement learning controller aims to learn the optimal fuzzy rules by reinforcement learning. In this approach, with respect to each fuzzy rule $r_{s,a}$, $V(r_{s,a})$ is defined to indicate the result of applying the rule. The optimal control policy can then be found by obtaining the V values of all fuzzy rules. The significance of $V(r_{s,a})$ is similar to $Q(s, a)$ in traditional reinforcement learning (Sutton & Barto, 1998). However, the Q-function used to update $Q(s, a)$ cannot be applied for updating $V(r_{s,a})$, because the states and actions in Q-function (5.3) are discrete and exclusive.

To address this problem, the triggers for updating $V(r_{s,a})$ have to be defined because it is hard to find "sharp" fuzzy state-transition time points using this approach. For example, when the target is 0.7 meters away, the membership value of fuzzy state "target found" is 0.1; when the target is 0.2 meters away, the membership value of fuzzy state "target found" is 0.9. In both cases, the fuzzy state "target found" is activated (non-zero), but with different membership values. Therefore, two triggers to update/reselect fuzzy rules are set as follows:

- The fuzzy state has a zero/non-zero change; or
- The fuzzy state has been activated (non-zero) for a long period of time (N simulation steps).

When either one of the above two conditions is satisfied, $V(r_{s,a})$ will be updated by Equation (5.5). In this equation, α is the learning rate; and *reward* is the feedback with respect to the progress of the mission. The new state, s_{t+1} in Equation (5.3) does not appear in this Equation (5.5) because the "next" fuzzy state is usually the same as the previous state although they may have different membership values, e.g., the fuzzy state "target found = 0.1" may be changed to fuzzy state "target found = 0.9". Therefore, it is not necessary to add the variable/value relating to the "next state".

$$V(r_{s,a}) \leftarrow (1-\alpha)V(r_{s,a}) + \alpha(reward) \tag{5.5}$$

In the fuzzy inference system, more than one fuzzy state may be activated at any one time; therefore, the output of the fuzzy inference system is the combination of fuzzy actions corresponding to these fuzzy states, as shown in Equation (5.4). In this case, the robot will learn more than one fuzzy rule concurrently. The value of each fuzzy rule is updated according to Equation (5.5).

After updating $V(r_{s,a})$, the learning controller needs to reselect fuzzy rules (fuzzy actions) for the robot to perform and test. To explore and exploit the possible fuzzy rules (actions), the controller has to add an exploration factor (random value) to each V value of the fuzzy rule. The fuzzy rules with the highest resultant values will then be chosen. It should be noted that this random factor is only used for fuzzy rules selection; it will not affect the actual V values of the fuzzy rules.

5.2.3.4 Coordination of Concurrent Learning Processes

The learning coordination algorithm is similar to Algorithm 5.4. If the value of the best action is much larger than the average of all actions in any one state, the robot stops learning in this state

and thereafter always chooses this best action in this state. In other words, a robot will fix its control policy when it has learned enough; future improvement of the group performance is then left to other learning robots.

5.2.3.5 Implementation of Tracking Problem

To implement the distributed learning controller for multi-robot tracking of multiple moving targets, two fuzzy states "target is found" and "target is tracked by others", and two fuzzy actions "track target" and "leave target", are defined as shown in Figure 5-8 and Figure 5-9. These fuzzy states and actions are designed based on human experiences.

As shown in Figure 5-8, the membership degree (value) of the fuzzy states indicates the degree of the state with respect to the distance to the target, or the distance between target and other robots. When the target is near the robot, the degree of fuzzy state "target is found" is large; when the target is tracked by another robot, the degree of the fuzzy state "target is tracked by others" is large if the target and other robot(s) are near to each other.

The membership degree (value) of the fuzzy action "track target" is the weight of the attractive forces to the target, i.e., $w_{R_i}^{T_j}$ in Equation (5.2). Higher membership degree indicates stronger preference to approach the targets. As shown in Figure 5-9, the degree of the fuzzy action "track target" is large when the target is far because the robot will lose the target if the tracking action is weak under this condition. The membership degree of the fuzzy action "leave target" is the inverse of the weight of the attractive forces. Higher membership degree indicates stronger preference to leave the targets. When the target is relatively near, the degree of the fuzzy action

"leave target" is large because the robot may hit the target if the leaving action is weak under this condition.

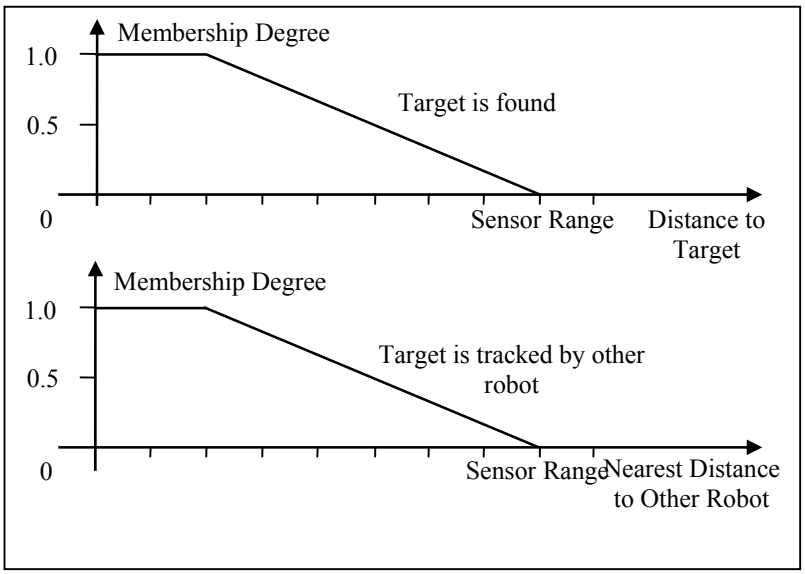

Figure 5-8 Definition of Fuzzy States

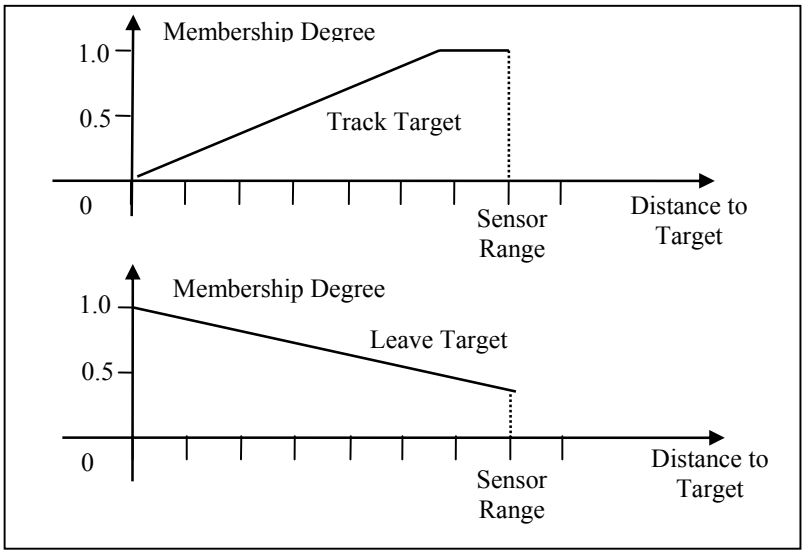

Figure 5-9 Definition of Fuzzy Actions

In reinforcement learning, one important issue is the generation of rewards because rewards represent the objectives of the designer and can directly affect the learning results. Since task-level cooperation is desired, the following behaviors should be encouraged: (i) track target; and

(ii) leave the target that is being tracked by other robots. For this purpose, this study defines three types of rewards:

- *Reward_TT/Reward_LT*: track/lose target reward (positive/negative) – awarded if the robot tracks/loses targets.
- *Reward_WT*: waste time reward (negative) – awarded if the robot tracks a target that is currently being tracked by other robots.
- *Reward_PT*: pass target reward (positive) – awarded if the robot passes the target to other robot(s) to track.

For each individual robot, the rewards for the distributed learning controller are generated by its local sensing. For example, if robots *A* and *B* are tracking the same target, from the perspective of robot *A*, it cannot generate the *Reward_TT,* and *Reward_WT* if it detects *B*.

5.2.3.6 Summary

In this subsection, a distributed fuzzy reinforcement learning controller which applies fuzzy logic to reinforcement learning is proposed. This controller enables robots to generate cooperative behaviors based on fuzzy states and actions. In addition, a nature-inspired distributed learning control algorithm is developed to coordinate the concurrent learning processes. The proposed algorithm could help avoid the generation of local sub-optimal control policies or the cyclic switching of control policies without requiring explicit intercommunications among robots.

5.2.4 Summary

One of the ultimate goals of robotics and artificial intelligence research is making use of machine learning methodologies to achieve multi-robot concurrent learning of cooperation. In this subsection, two distributed reinforcement learning controllers are proposed for concurrent learning. The distributed controllers can enable the robots to generate cooperative behaviors based on local sensing and without the need for direct intercommunications. In addition, a nature-inspired distributed learning control algorithm is developed to coordinate the concurrent learning processes. These algorithms are simple and scalable, and can be applied in most real applications.

5.3 Simulation Tests and Discussions

In this chapter, the following deliberatively designed controllers and learning controllers for multi-robot tracking of multiple moving targets are proposed:

- Potential field-based tracking with adaptive (conditional) weight change – selective adjustment of pure potential field-based control
- Reinforcement Learning with behavior-based control network
- Fuzzy reinforcement learning

To demonstrate the efficiency of the proposed control algorithms (controllers), the following algorithms are used as the benchmark for comparison:

- Pure potential field-based tracking
- Potential field-based tracking with weight change – all-adjust weight change

Because the research aim of multi-robot cooperative target tracking is to maximize the number of observed targets and minimize the number of robots needed to track targets, the following two metrics are used to evaluate the performance of the multi-robot systems:

- Average number of tracked targets (*Ave_T*) – Higher value of *Ave_T* corresponds to better performance. Let T_i denote the number of targets being tracked at simulation step *i*, and *N* denote the total number of steps of a simulation run; then $Ave\_T = \frac{1}{N}\sum_{i=1}^{N} T_i$.
- Average number of robots needed to tracked one target (*Ave_R*) – Lower value of *Ave_R* corresponds to better performance. Let R_i denote the number of robots that are needed to track one target at simulation step *i*, and *N* denote the total number of steps of a simulation run; then $Ave\_R = \frac{1}{N}\sum_{i=1}^{N} R_i$.

The learning performance of the two learning controllers is evaluated by the metric *Learning_Progress*:

- For the learning controller with a behavior-based control network, *Learning_Progress = Number of correctly learned weights / number of all weights that can be learned*. Here, "*number of all weights that can be learned*" is equivalent to the total number of states because one suitable action (weight) is needed for each state; and "*Number of correctly learned weights*" refers to the number of appropriate weights that are learned (the weight with the highest *Q* value is considered as the learned weight for that state) at that moment.
- For the learning controller with fuzzy logic, *Learning_Progress = Number of correctly learned fuzzy rules / number of all fuzzy rules that can be learned*. Here, "*number of all fuzzy rules that can be learned*" is equivalent to the total number of fuzzy states because one suitable fuzzy rule is needed for each fuzzy state; and "*Number of correctly learned*

fuzzy rules" refers to the number of appropriate rules that are learned (the rule with the highest *V* value is considered as the learned rule for that state) at that moment.

For both learning controllers, it is not easy to determine whether a weight (or rule) is appropriate or not (this is indeed the purpose of learning); therefore it is difficult to determine "*Number of correctly learned weights*" or "*Number of correctly learned fuzzy rules*". For example, for learning with behavior-based control networks, it is hard to analyze or calculate for state (*2, 1*) – two neighbor targets and one neighbor robot, which action (weight) is the optimal one for tracking. Therefore, in this study, the appropriate learning results are the ones that can generate the best tracking performance, i.e., the learning results of the best performing controller that can achieve the highest *Ave_T* and the lowest *Ave_R*.

Normally, the *Learning_Progress* is a value which increases with time. It shows the speed at which robots can learn the desirable behaviors. At simulation step *i*, each robot will have its *Learning_Progress*; therefore the presented *Learning_Progress* is the average value of the *Learning_Progress* of all robots.

5.3.1 Simulation Environment and Settings

The simulation environment is set as the following:

- The environment and robot is simulated by Webots, a well-known robot simulator (Olivier, 2004).
- As the focus of the test is to compare the tracking performance, especially in the context of cooperation, all irrelevant factors are eliminated, e.g., internal obstacles are removed

from the simulation scenario. However, the robots and targets are still objects that should not collide with one another.

- The initial locations of targets and robots are randomly assigned.

Other settings are as follows:

- Environment: 40 x 40m ~ 60 x 60m square plain area with small obstacles. The simulated robots and targets are less than 1m in diameter.
- For each control mode, the average of the simulation results of 10 simulation runs is used. Each run is 10000 simulation steps long. Each simulation step is about 0.1s long in real time.
- For the all-adjust heuristics of pure potential field-based control, if two or more robots find the same target as well as each other, they will all decrease the weight of the attractive force to the target. In the simulation, 5 different All-Weight Decrease Ratios (AWDR) are tested: 0.1, 0.3, 0.5, 0.7 and 0.9.
- For the selective-adjust heuristics of pure potential field-based control, if two or more robots find the same target as well as each other, the more distant robot(s) to the target will decrease the weight of the attractive force to the target. In the simulation, 5 different Selective-Weight Decrease Ratios (SWDR) are tested: 0.1, 0.3, 0.5, 0.7 and 0.9.

The settings of the learning controller with behavior-based control network are as follows:

- The initial Q value of all state-actions is 10.
- *Reward_TT* = 0.005 * track target time.
- *Reward_NR* = - 0.01 * near robot time.
- *Reward_SC* = $(m-a)*0.5 - (n-b) * 2.0$ (m and n are the current target and robot number; a and b are the previous target and robot number).

- *Reward_WT* = 0.1 * waste steps (waste steps are the simulations steps during which two or more robots track the same targets).
- Possible weights to learn are 0.1, 0.3, 0.5, 0.7, and 0.9.
- In each state, if the value of one action is above the average of the *Stop_Learning_Threshold*, the robot should stop learning for this state. In the simulation, two *Stop_Learning_Threshold* values are tested: 25% and infinite (no learning coordination).
- If the state remains unchanged for $N = 100$ simulation steps, the robot reselects the action. Usually, the robot is only allowed to change action when state changes. However, in the proposed learning system, the state does not change frequently. To increase the learning speed, the robot is designed to re-select action if it has been in the same states for a long time. This is depicted in Figure 5-5.
- When selecting an action, a random number uniformly distributed on the interval [-1, 1] is added to the real state-action value as the exploration factor. It should be noted that the exploration factor is used only for action selection, but not for updating the state-action value.

The settings of the fuzzy learning controller are as follows:

- The initial V values of all fuzzy rules are 10.
- *Reward_TT* /*Reward_LT* = 0.8/-0.8
- *Reward_WT* = -1.5
- *Reward_PT* = 2.0.
- For each state, if the value of one action (rule) is above the average of the *Stop_Learning_Threshold*, the robot should stop learning for this state. In the simulation,

two *Stop_Learning_Threshold* values are tested: 15% and infinite (no learning coordination).

- If the fuzzy state remains unchanged (non-zero) for $N = 50$ simulation steps, the robot updates and reselects the fuzzy rules.
- When selecting a fuzzy rule (action), a number uniformly distributed on the interval [-1, 1] is added to the real value of the fuzzy rule as the exploration factor. This exploration factor is used only for fuzzy rule selection, but not for updating the value of the fuzzy rule.

5.3.2 Simulation Results and Discussion

In this subsection, the following results are compared and presented:

- Performance of tracking. The performances of the proposed selective-adjust heuristic of pure APF and the two learning controllers (with the learning coordination algorithm) are compared with the pure potential field-based controller and the all-adjust heuristic of pure APF, which have been proposed by other researchers in related work.
- Performance of learning. In addition to traditional reinforcement learning, this study develops a learning coordination algorithm to improve the learning performance. This study compares the learning performance of the controller with and without the proposed coordination algorithm.

5.3.2.1 Tracking Performance

In the following, the average number of tracked targets is shown in Figure 5-10 to Figure 5-14. In each of these figures, the left subfigure shows the performance of the homogeneous robot team (all

robots are identical) and the right subfigure shows the performance of the heterogeneous robot team (one of the robots is 30% faster than others). Usually for homogeneous robot team, the robots may learn similar behaviors (or similar combination of elementary behaviors) because their functionality is the same. However, for heterogeneous robot team, the robots may obtain different behaviors because they have different capabilities. It is meaningful to run the proposed learning algorithms for both homogeneous and heterogeneous robots to test the performance of learning. For ease of comparison, all the values of *ave_T* are normalized such that the highest value is 1. (Before normalization, the *ave_T* of pure APF in 1T2R, 3T3R, 3T6R, 6T3R and 6T6R scenarios is 1, 2.58, 2.95, 3.44, and 4.68, respectively.)

In each subfigure of Figure 5-10 to Figure 5-14, the leftmost column is the performance of pure potential field-based control; the second and third columns are the performances of controllers with all-adjust heuristic and selective-adjust heuristic, respectively. The fourth column is the performance of the learning controller with a behavior-based control network, and the rightmost column is the performance of the learning controller integrated with fuzzy logic.

In all simulation runs, the settings of the pure potential field-based controller and the two learning controllers are unchanged, i.e., the weight values (for pure potential field-based control) and the learning parameters (e.g., learning rate for proposed learning controllers) are constant. Therefore, regarding each of these three controllers, the performance shown in the subfigures is a single value obtained from the average of all simulation runs.

On the other hand, in the simulation, different parameter settings for the all-adjust heuristic and selective-adjust heuristic controllers are tested, i.e., the AWDR (all-adjust heuristic) and SWDR (selective-adjust heuristic) are changed in different simulation runs. One performance value is obtained for each parameter setting. Therefore, each subfigure shows the values for the highest,

lowest and average performance of all 5 settings of these two controllers. For example, the left subfigure of Figure 5-11 shows three performance values for the all-adjust heuristic controller: the lowest is 0.934 (when AWDR equals 0.1), highest is 0.980 (when AWDR equals 0.7), and the average value is 0.954.

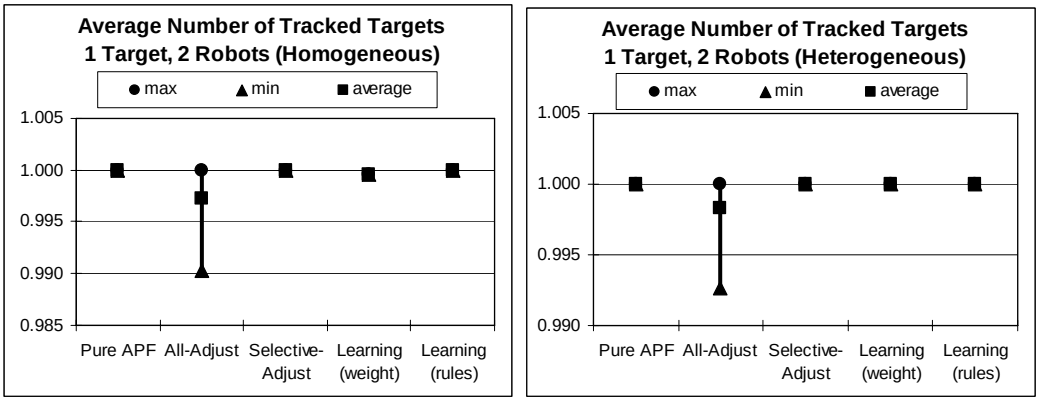

Figure 5-10 Average Number of Tracked Targets - One Target, Two Robots
(left: homogeneous robot team; right: heterogeneous)

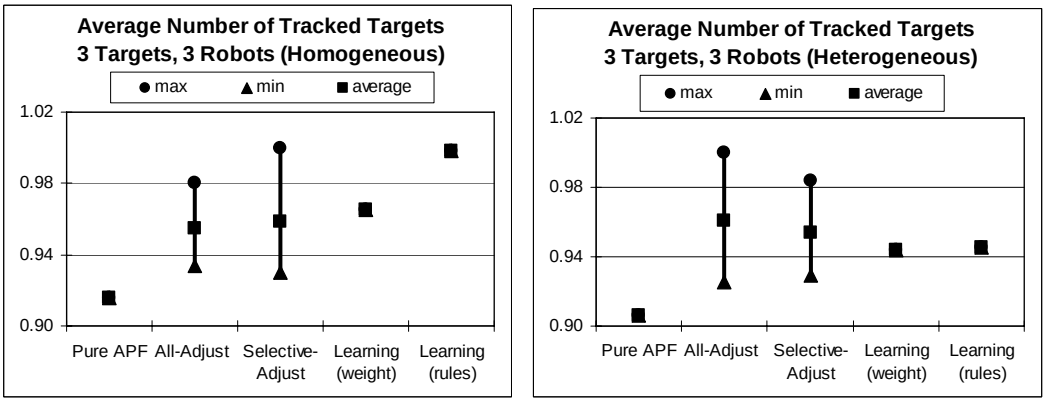

Figure 5-11 Average Number of Tracked Targets - Three Targets, Three Robots
(left: homogeneous robot team; right: heterogeneous)

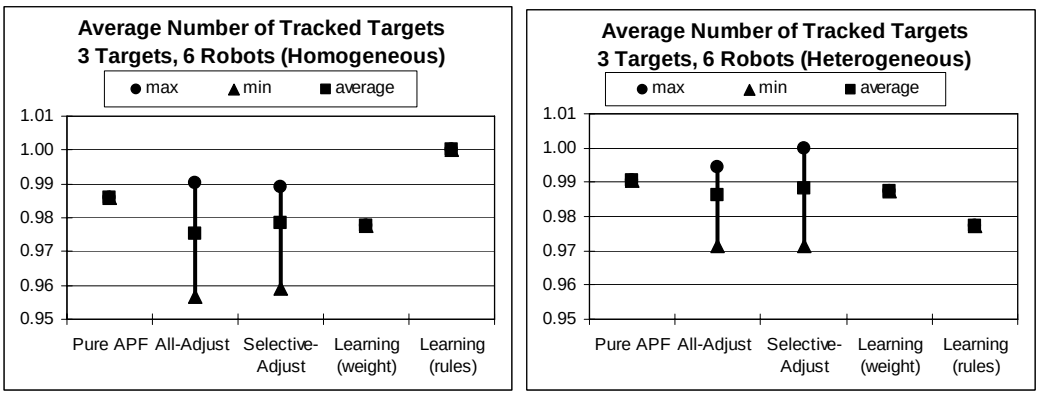

Figure 5-12 Average Number of Tracked Targets - Three Targets, Six Robots
(left: homogeneous robot team; right: heterogeneous)

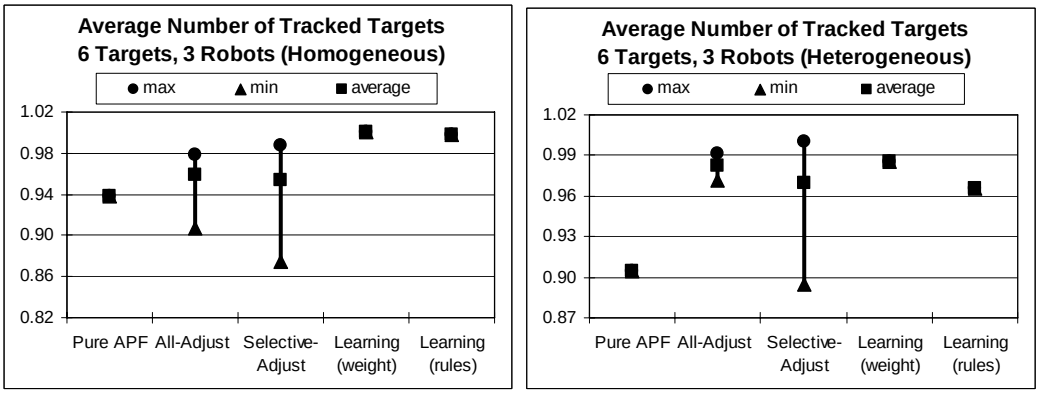

Figure 5-13 Average Number of Tracked Targets - Six Targets, Three Robots
(left: homogeneous robot team; right: heterogeneous)

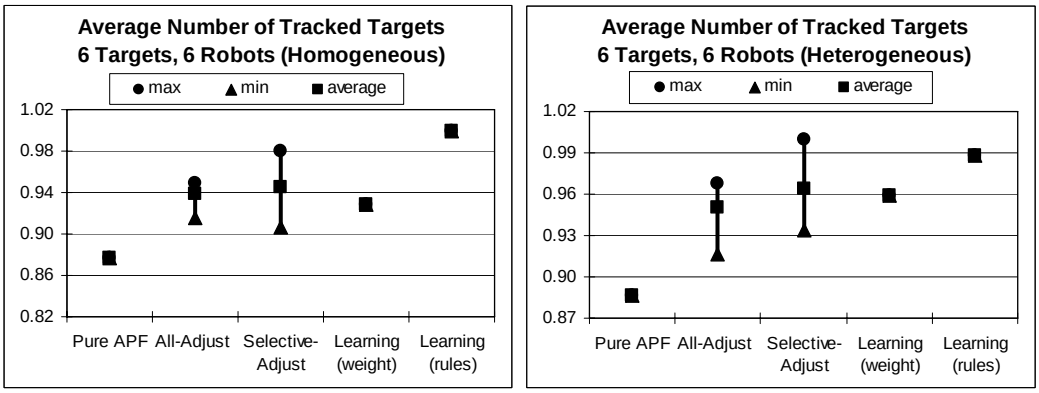

Figure 5-14 Average Number of Tracked Targets - Six Targets, Six Robots
(left: homogeneous robot team; right: heterogeneous)

From these figures, we can see that in most cases, the pure potential field-based controller yields the worst performance. This is mostly due to the problem described in Section 5.1.1. The all-adjust

heuristic can enable the robots to give up the target(s) that is being tracked by others. Therefore it can improve the cooperation among robots to better utilize the robot resource. Its performance is usually better than the pure potential field-based control. However, an inappropriate All-Weight Decrease Ratio (AWDR) may generate undesirable results: if AWDR is too small (e.g., 0.1), all robots may leave the target; if AWDR is too large (e.g., 0.9), none of the robots may leave the target. Because of this, sometimes the all-adjust heuristic works even worse than the pure APF. In simulations, the AWDR is selected as 0.1, 0.3, 0.5, 0.7, and 0.9. When AWDR equals 0.9, the tracking performance should be close to pure potential field-based tracking, which is the same as all-adjust heuristic with AWDR=1.0. However, the above figures show that sometimes the performance of pure potential field-based tracking is quite different from the all-adjust heuristic. This indicates that the tracking performance is quite sensitive to the AWDR value.

Comparing to the all-adjust heuristic, the selective-adjust heuristic is more intelligent because it can let the robots select the most suitable robot (the one nearest to the target) to track the target, and thus the problem associated with the all-adjust heuristic is less likely to happen. The results show that it usually performs better than the all-adjust heuristic. However, if the Selective-Weight Decrease Ratio (SWDR) is inappropriate, it may badly influence the performance. For example, when SWDR is very small (e.g., 0.1), the further robot(s) may leave the target immediately, but in some scenario (where has more robots than targets) this is not very necessary and will increase the possibility that the robots lose the target; on the other hand, when SWDR is very large (e.g., 0.9), the further robot(s) may not leave the target.

The two heuristics can improve the performance; however, the improvement is not consistent. They may perform better or worse than the pure potential field-based control, e.g., Figure 5-13. This is because the hardcoded controllers are sensitive to the parameter settings, and the optimal parameter setting varies for different scenarios. For example, as shown in Figure 5-15, for the all-

adjust heuristic of potential field-based control, the different parameter (AWDR) settings have quite different results in different scenarios. In the scenario "3 targets and 6 robots" with a homogeneous robot team, AWDR=0.7 achieves the best performance, while in scenario "6 targets and 3 robots" AWDR=0.1 achieves the best performance.

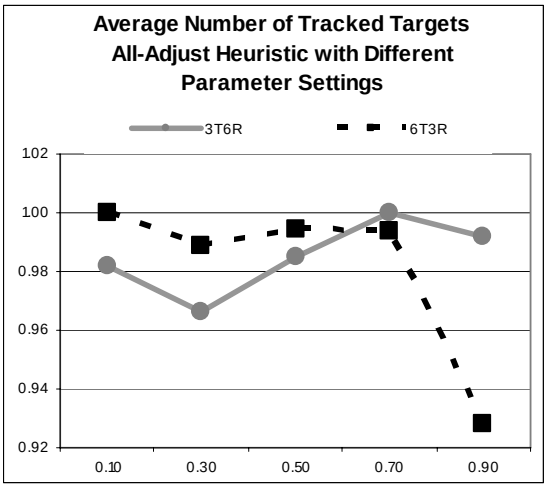

Figure 5-15 Performance with Different Parameter Settings

In contrast to the hardcoded controllers, which require tuning to achieve optimal control, the learning controller does not need to select any important parameters. The system is able to maintain good performance in all cases as shown in Figure 5-10 to Figure 5-14:

- The proposed learning controllers have the ability to adapt to different scenarios. They are especially powerful in scenarios that are comprised of more robots and targets. This is possibly because the learning controller can easily handle the complexity of the system, while it is difficult to find an optimal parameter for such scenarios using the hardcoded method.
- In both homogeneous and heterogeneous robot teams, the performance of the learning robots is satisfactory and consistent. This again shows the adaptivity of the learning controller.

Comparing the two learning controllers, we find that the fuzzy learning controller works better than the learning controller with behavior-based control network. This is possibly due to the fact that the fuzzy learning controller considers the relative distance between robots and targets, while the other learning controller does not utilize such information.

From the simulation results, we find that for both homogeneous and heterogeneous robot teams, the proposed heuristics and the learning controller could improve the system performance. These further justified the efficacy of the proposed tracking algorithms. The simulation also shows that all tracking algorithms are consistent in performance. As shown in Figure 5-16, at the most complex scenario containing 6 robots and 6 targets, all tracking algorithms has very small standard deviation in *ave_T*.

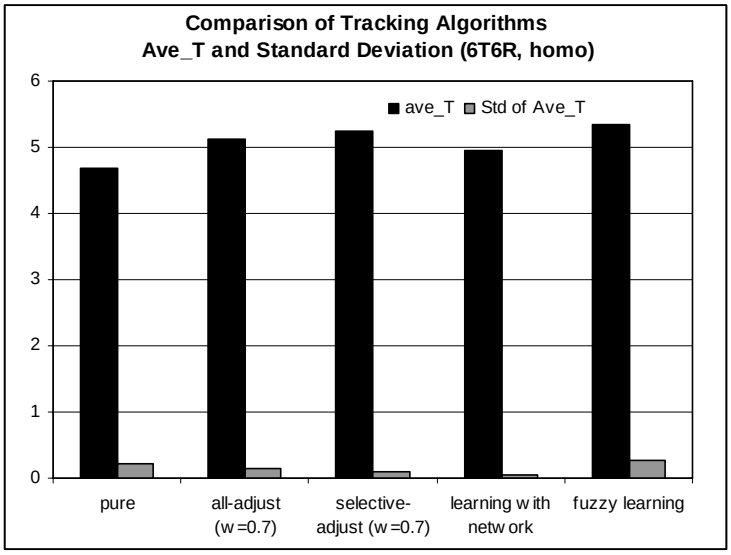

Figure 5-16 Standard Deviation of *Ave_T*
(6 targets and 6 robots, homogeneous robots)

In cooperative tracking, in addition to the number of tracked targets, another important performance metric is the average number of robots that are needed to track one target (*ave_R*). If

the robot team is highly cooperative, fewer robots are required to track one target, resulting in small *ave_R* values.

During the simulation, at each step, the number of tracking robots (say TR) and the number of tracked targets (say TT) are recorded. The *ave_R* is the average of TR/TT throughout simulation (if TT=0, let TR/TT=0). For example, if two robots always track one target at all the time, the *ave_R* is 2 (=2/1). The average number of robots that are needed to track targets (*ave_R*) is shown in Figure 5-17 to Figure 5-21. In these figures, the values of *ave_R* are normalized such that the highest one is 1 in each figure. (Before normalization, the *ave_R* of pure APF in 1T2R, 3T3R, 3T6R, 6T3R and 6T6R scenarios is 1.99, 1.15, 1.99, 0.89, and 1.22, respectively.)

The results of *ave_R* are consistent with the results of *ave_T*: the pure APF performs the worst; the all-adjust and selective-adjust heuristics are sensitive to the parameter settings; and the learning controllers are robust and adaptive to changes in the environment.

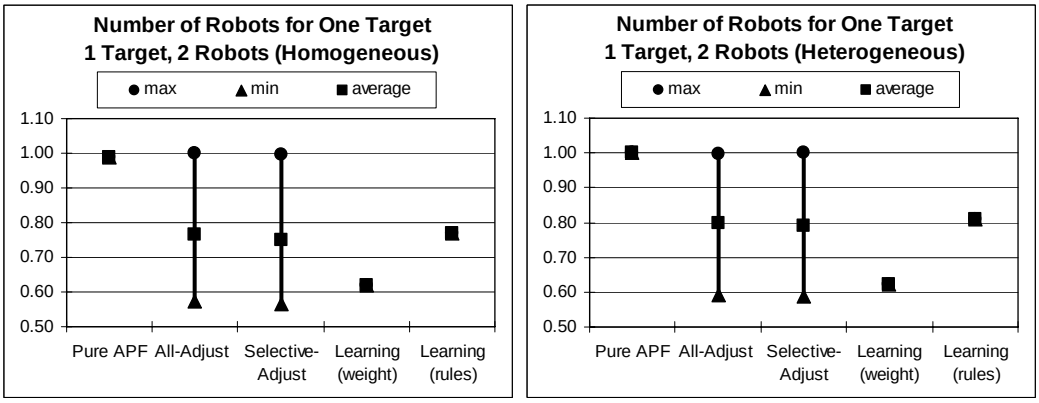

Figure 5-17 Number of Robots for One Target - One Target, Two Robots
(left: homogeneous robot team; right: heterogeneous)

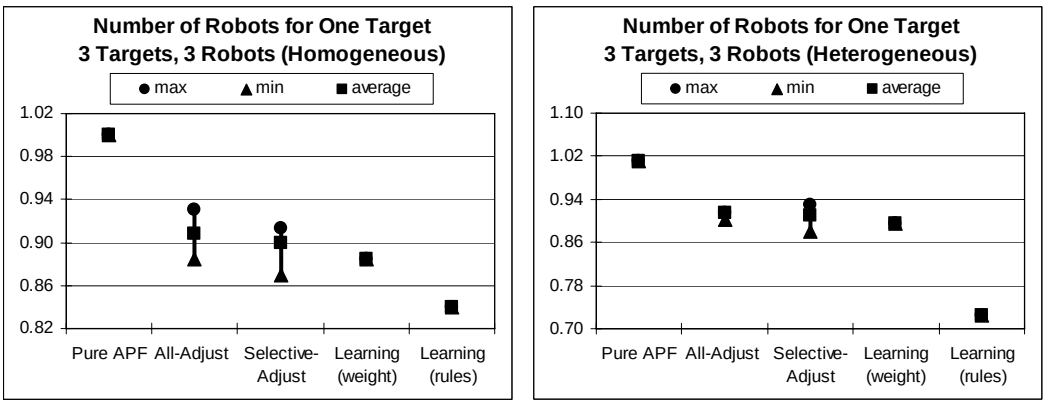

Figure 5-18 Number of Robots for One Target - Three Targets, Three Robots
(left: homogeneous robot team; right: heterogeneous)

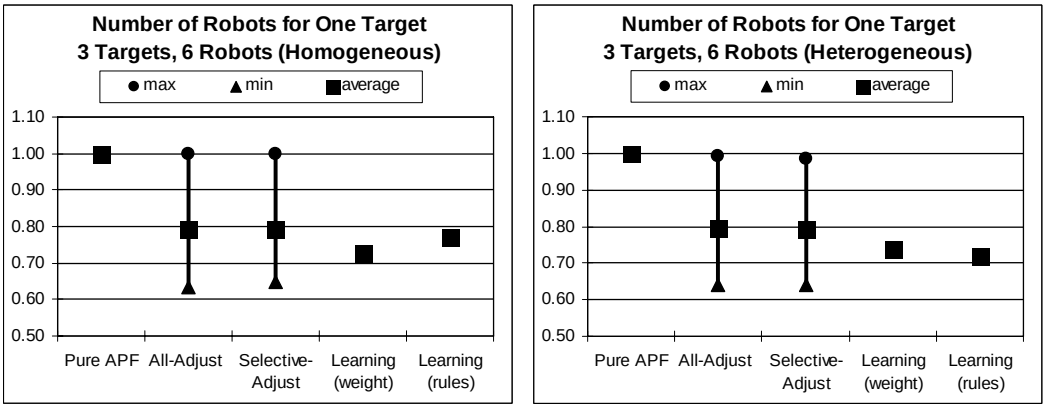

Figure 5-19 Number of Robots for One Target - Three Targets, Six Robots
(left: homogeneous robot team; right: heterogeneous)

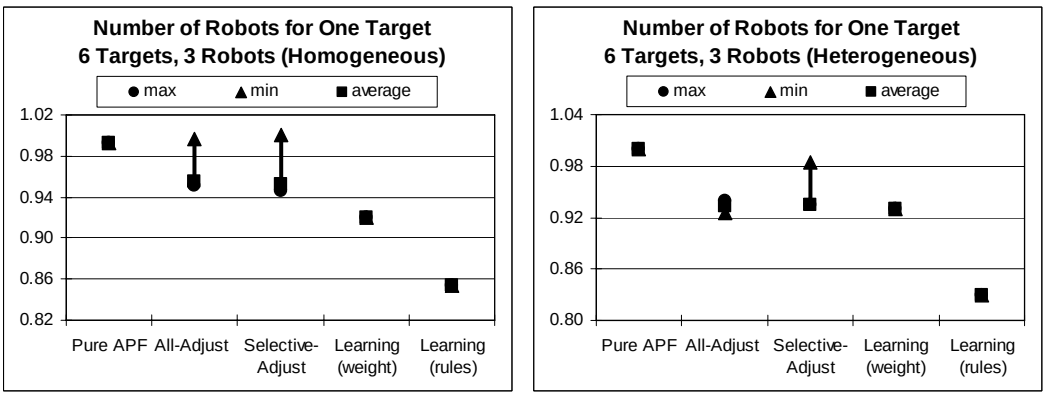

Figure 5-20 Number of Robots for One Target - Six Targets, Three Robots
(left: homogeneous robot team; right: heterogeneous)

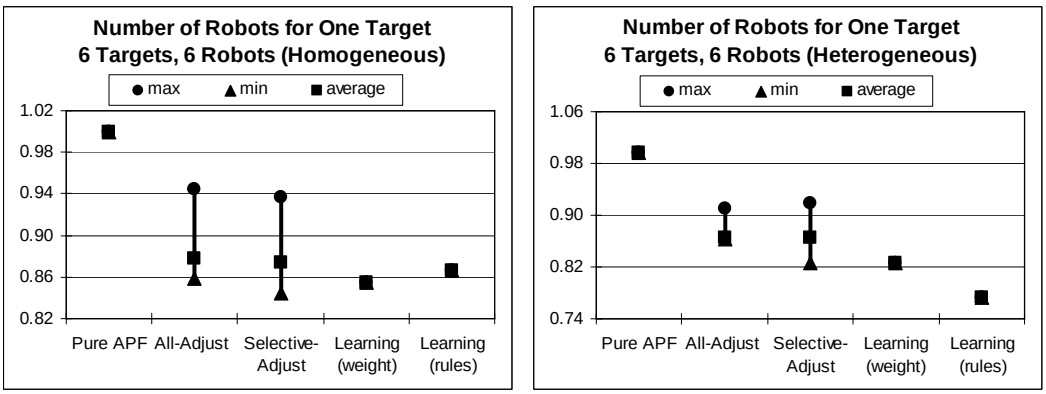

Figure 5-21 Number of Robots for One Target - Six Targets, Six Robots
(left: homogeneous robot team; right: heterogeneous)

From the simulation results, we see that the performance of the learning controller is sometimes worse than the heuristics of the pure potential field-based control (especially the selective-adjust heuristic). This may be due to the following reasons:

- The learning controller may take some time to learn the optimal control policy and strategy. During the initial part of the learning process, the robots have to try all possible behaviors or rules; therefore they may behave inappropriately and deteriorate the performance of tracking.
- For multi-robot concurrent learning, the contention among robots may influence the learning performance. The robots may learn sub-optimal control strategies.
- The selective-adjust weight heuristic considers the distance between robots and targets, but the behavior-based control network does not utilize this information. This is because we try to avoid the increase in the input state space and allow the learning to converge within satisfactory time.

Videos and photos have been taken for all simulations. Please kindly refer to http://guppy.mpe.nus.edu.sg/~mpeangh/kevin to review them to have a better understanding of the system.

5.3.2.2 Analysis of Concurrent Learning Processes

For the tracking problem, the research aim is to maximize the number of observed targets and minimize the number of robots needed to track targets. To achieve this, the robot team needs to be fully utilized such that the robots should track detected targets and also try to find undetected targets. For example, when two robots find the same target, they should behave differently such that one continues to track and the other leaves to search for other targets. This is the main aim of the learning controllers. The results shown in Section 5.3.2.1 have demonstrated the efficacy of the proposed advanced potential field-based controller (selective-adjust heuristic) and the learning controllers.

While the learning methods can improve the adaptivity of the robot systems, they may be affected by the interference among concurrently learning robots. To eliminate the influence of such interference, a distributed learning coordination algorithm is developed in this study. When a robot has learned enough for one state, it may stop learning for this state and use the learned control policy as the optimal control policy. To test the efficacy of the proposed learning coordination algorithm, the learning process of the controllers (with and without coordination) is examined.

In Figure 5-22 to Figure 5-26, the learning processes of the behavior network based learning controller (with/without coordination) are compared (The "I" in the figure caption means controller 1 – learning controller with a behavior-based control network). In these figures, the learning progress is normalized such that the best learning result is 1. From these figures, we can see that with the increase in time, the learning controller can gradually find the optimal control policy for the robots. More importantly, the learning controller with coordination can achieve better learning results (6 out of 10) than without the coordination algorithm (4 out of 10).

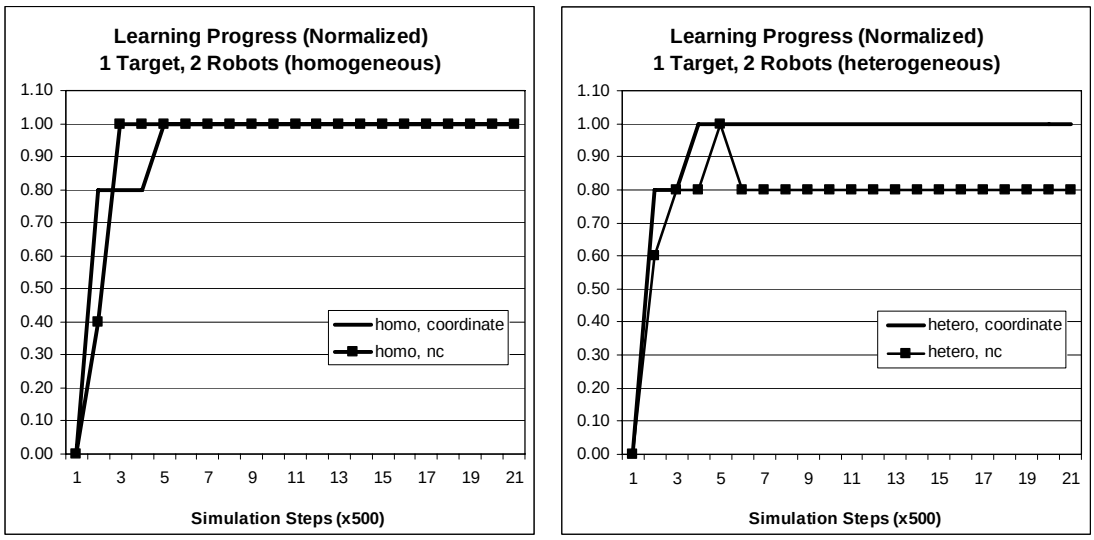

Figure 5-22 Learning Progress (normalized) - I - One Target, Two Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

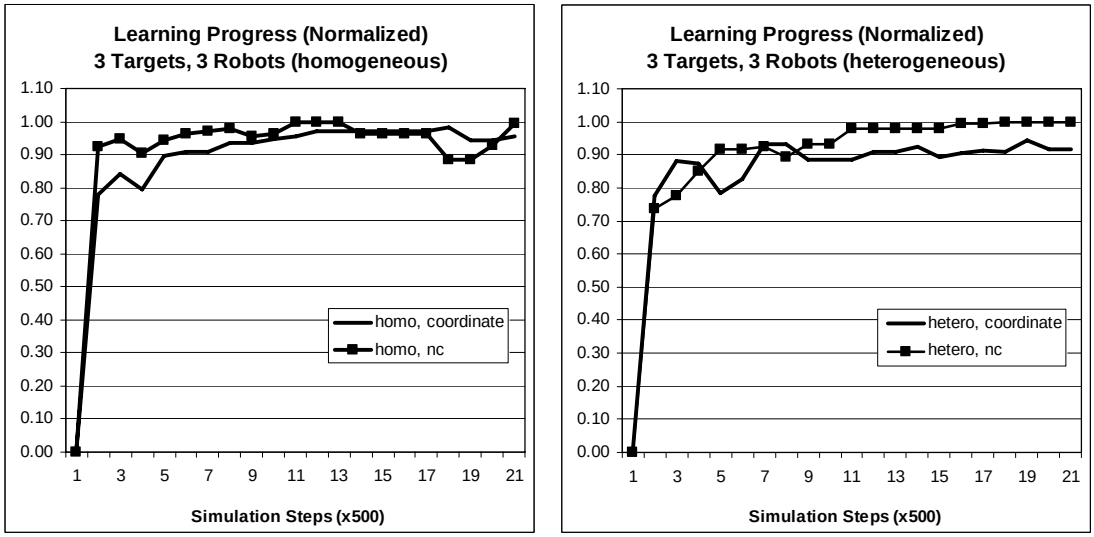

Figure 5-23 Learning Progress (normalized) - I - Three Targets, Three Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

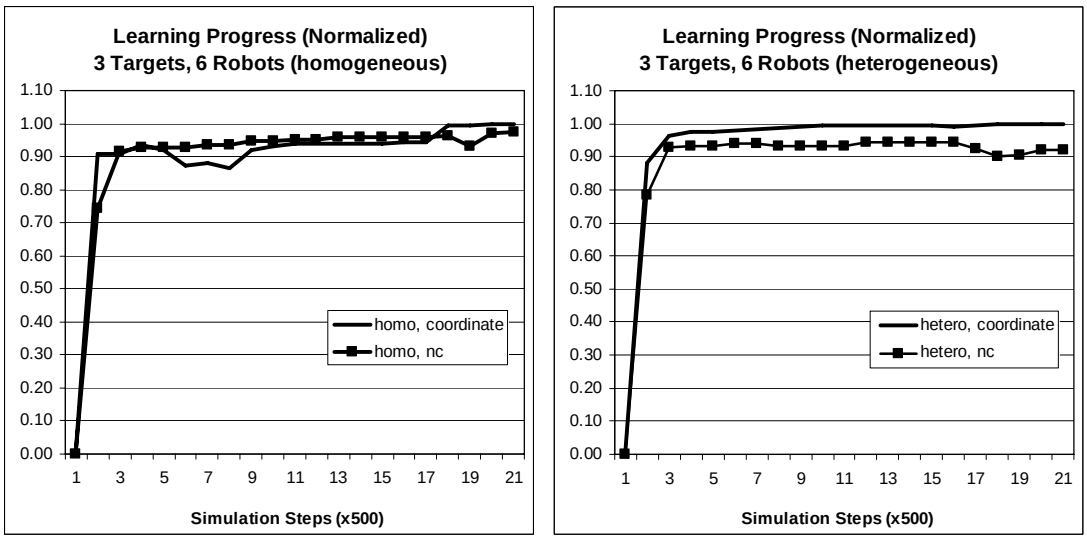

Figure 5-24 Learning Progress (normalized) - I - Three Targets, Six Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

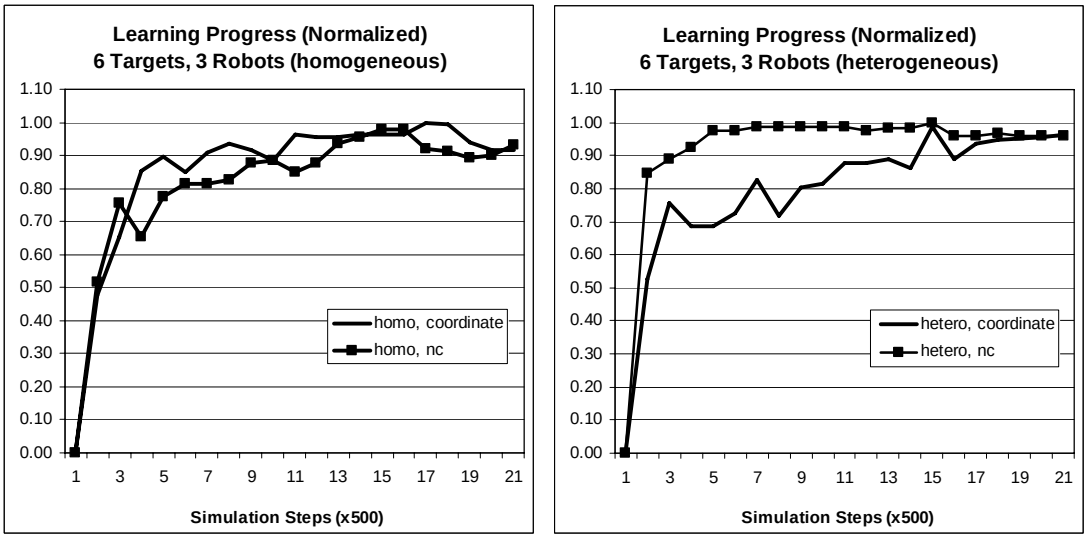

Figure 5-25 Learning Progress (normalized) - I - Six Targets, Three Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

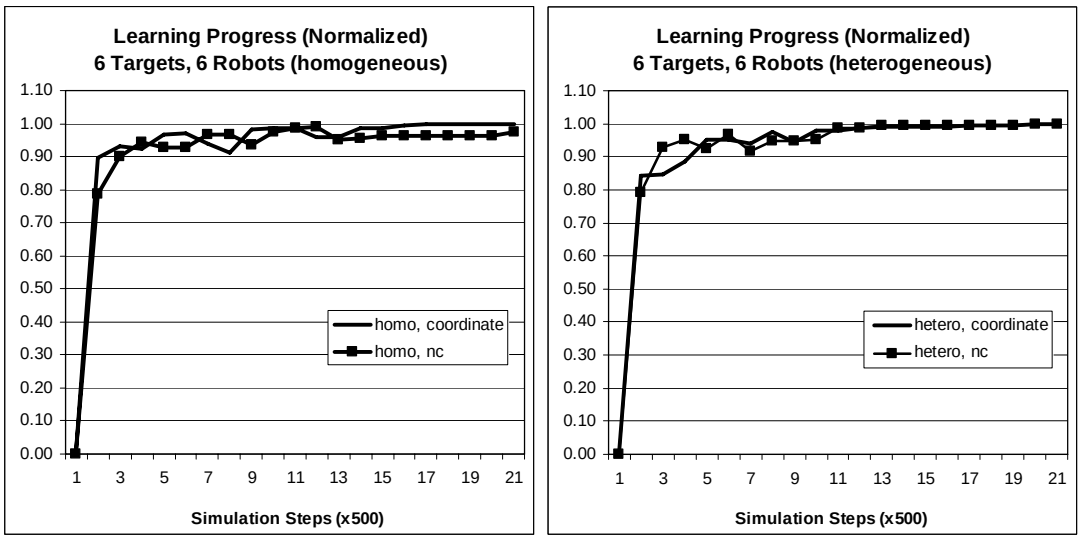

Figure 5-26 Learning Progress (normalized) - I - Six Targets, Six Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

From Figure 5-22 to Figure 5-26, we find that the advantage of the learning coordination algorithm is not very apparent. However, because the coordination of learning may reduce the interference among robots, and thus let the robots have less "struggle" in tracking targets, the robots have better tracking performance. This is shown in Figure 5-27 and Figure 5-28. The learning robots that have learning coordination are able to track more targets in average.

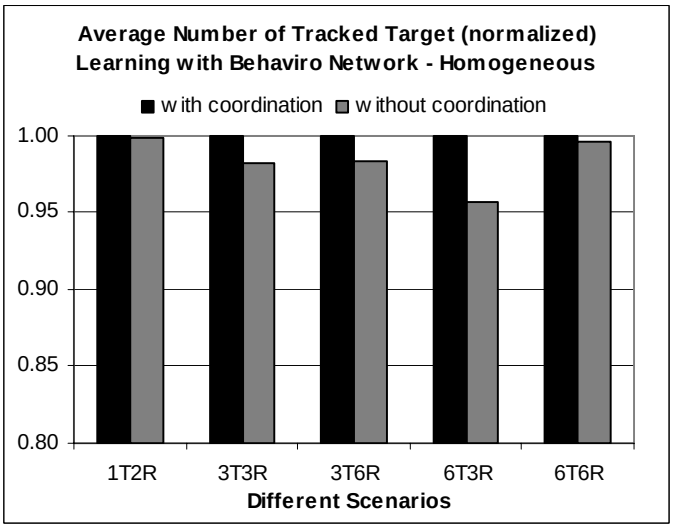

Figure 5-27 Tracking Comparison - Learning with and without Coordination
Learning with Behavior-based Network (homogeneous team)

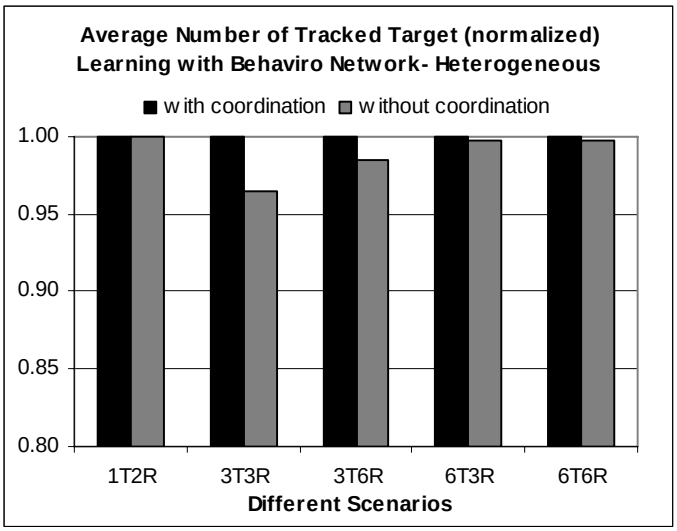

Figure 5-28 Tracking Comparison - Learning with and without Coordination
Learning with Behavior-based Network (heterogeneous team)

In Figure 5-29 to Figure 5-33, the learning processes of the fuzzy learning controllers (with/without coordination) are compared (The "II" in the figure caption means controller 2 – fuzzy reinforcement learning controller). In these figures, the learning progress is normalized such that the best learning result is 1. From these figures, we can see that with the increase in time, the learning controller can gradually find the optimal control policy for the robots. More importantly, the learning controller with coordination can achieve much better learning results (9 out of 10) than without the coordination algorithm (1 out of 10).

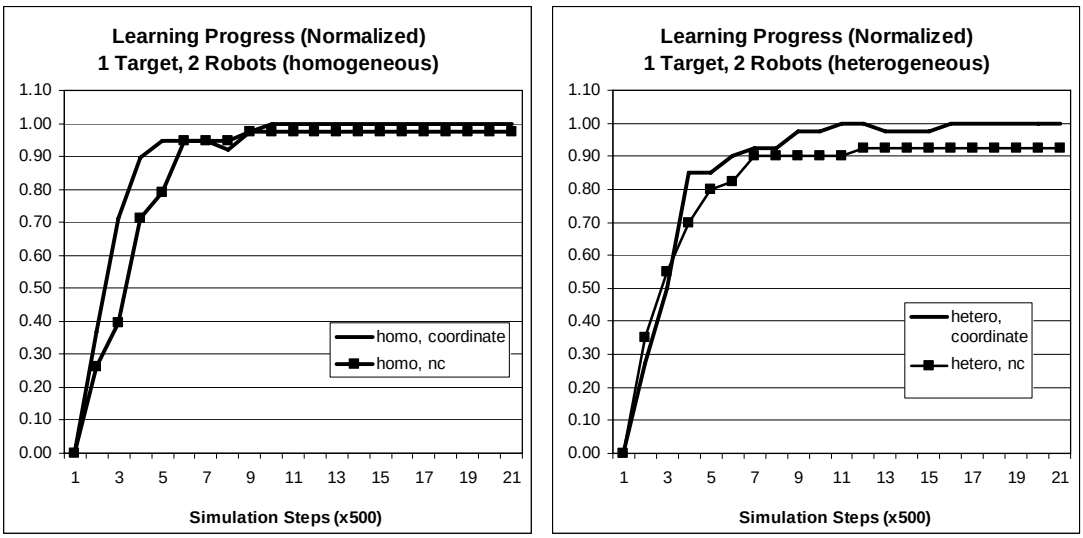

Figure 5-29 Learning Progress (normalized) - II - One Target, Two Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

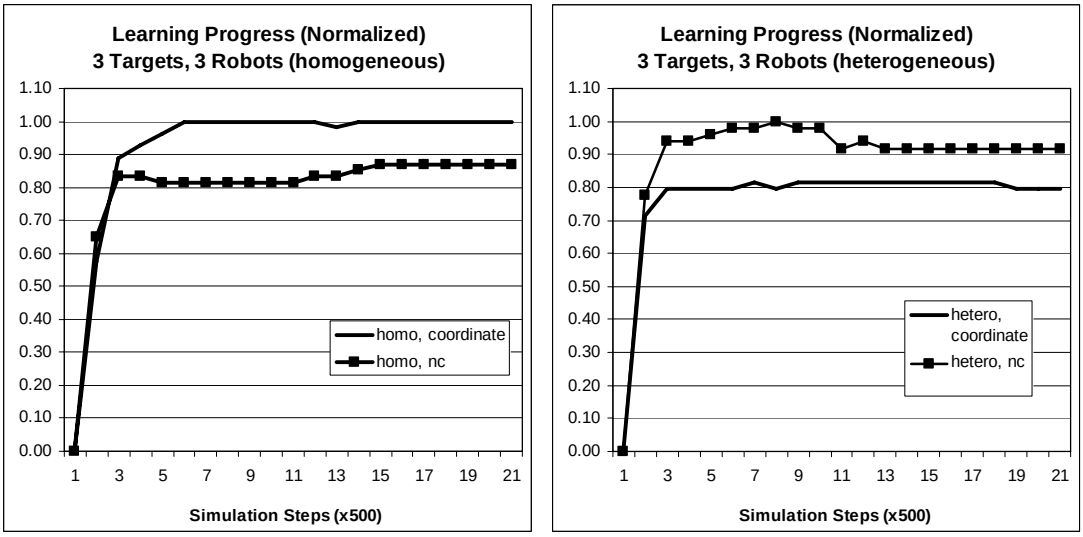

Figure 5-30 Learning Progress (normalized) - II - Three Targets, Three Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

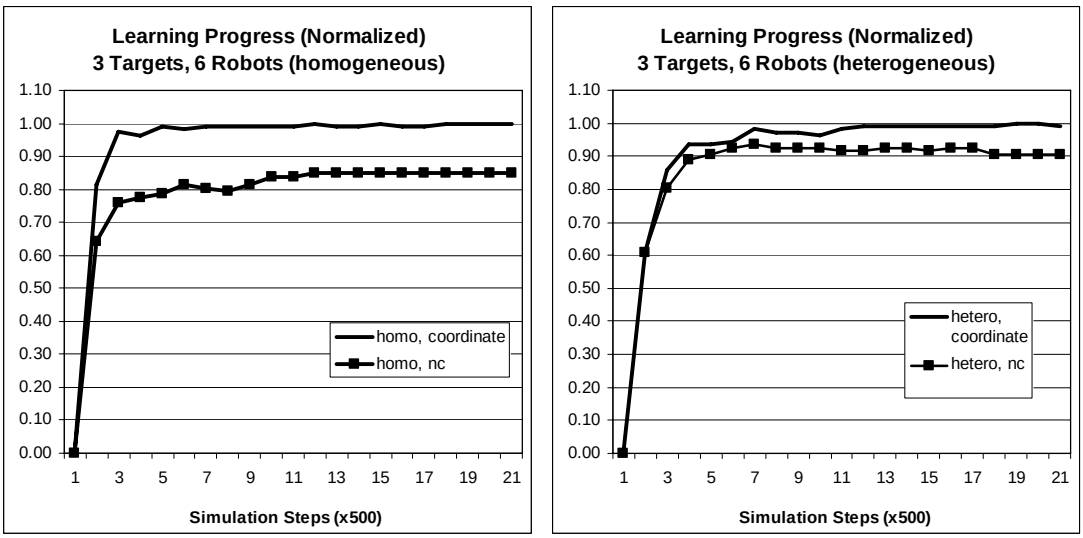

Figure 5-31 Learning Progress (normalized) - II - Three Targets, Six Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

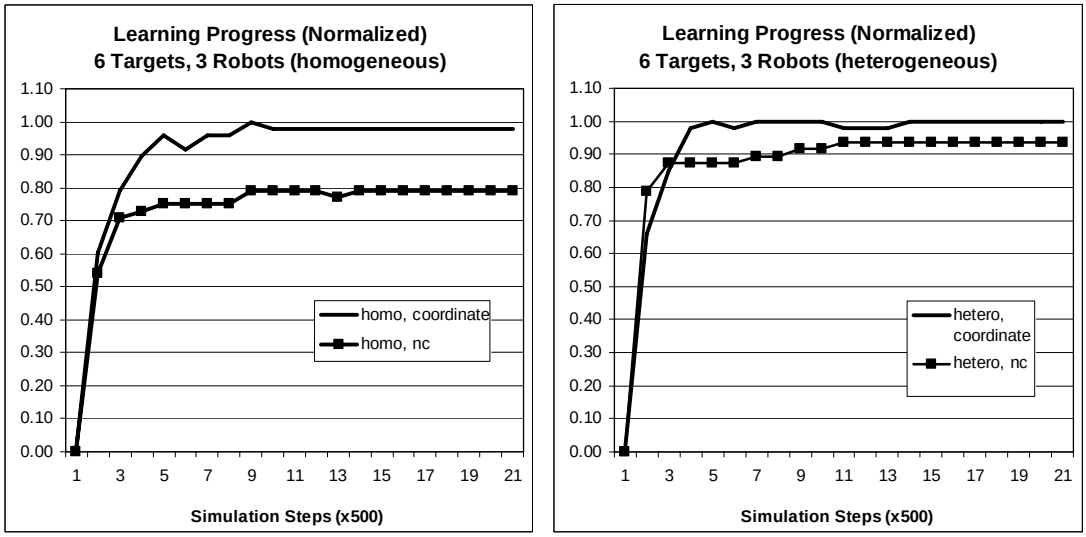

Figure 5-32 Learning Progress (normalized) - II - Six Targets, Three Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

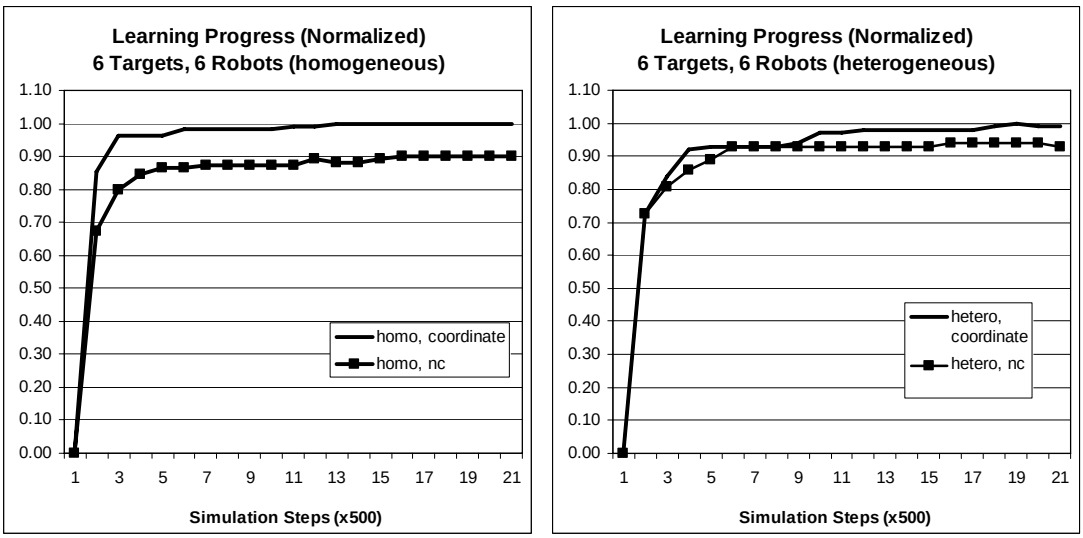

Figure 5-33 Learning Progress (normalized) - II - Six Targets, Six Robots
(left: homogeneous robot team; right: heterogeneous;
coordinate: with the proposed coordination; nc: without the coordination)

In addition to the learning process shown in Figure 5-29 to Figure 5-33, we also find that the learning coordination algorithm is able to let the robots track more targets. This is shown in Figure 5-34 and Figure 5-35. The learning robots that have learning coordination are able to track more targets in average.

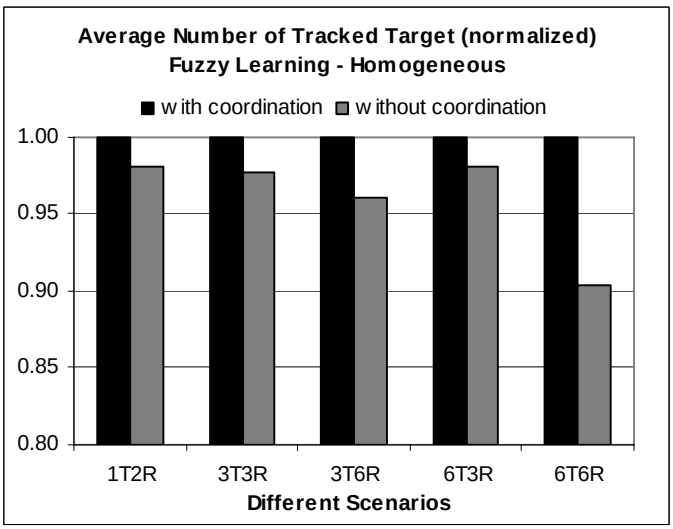

Figure 5-34 Tracking Comparison - Learning with and without Coordination
Fuzzy Learning (homogeneous team)

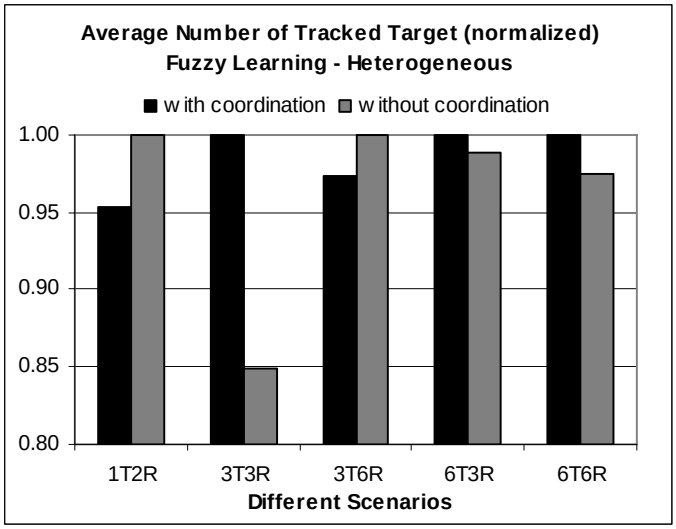

Figure 5-35 Tracking Comparison - Learning with and without Coordination
Fuzzy Learning (heterogeneous team)

While the learning coordination algorithm can improve the learning performance, the learning controller may not always generate the optimal control policy. It sometimes could generate worse performance as compared to the learning controller without any coordination. A possible explanation for this phenomenon is that the optimal stop-learning thresholds are different for varying learning scenarios: small threshold values may be suitable for small robot groups, while large thresholds may be suitable for larger robot groups. This difference in optimal threshold values is due to the fact that large robot groups tend to have more interference; therefore they require large stop-learning thresholds to eliminate the sub-optimal behaviors (or rules). However, if the thresholds are too large, the robots may "hesitate" to fix the good behavior (or rules); therefore concurrent learning robots may experience more "struggles". It is of paramount importance to find the optimal stop-learning thresholds as part of future study.

In addition, we find that the learning coordination algorithm achieves better results for the fuzzy reinforcement learning controller (9:1) than the behavior learning controller (6:4). This may be due to the following reasons:

- The proposed learning coordination algorithm tries to coordinate the concurrent learning processes that are learning for the same state (e.g., the corridor encounter case explained in Section 5.2.2.5). If the robots are in the same state, this learning coordination algorithm may effectively improve the learning. On the other hand, if the robots are less likely to be in the same state, this coordination algorithm may not work well.
- In the fuzzy reinforcement learning controller, there are only two states; however, in the learning controller that integrates with the behavior control network, there are $m*(n+1)$ states (m is the number of robots, n is the number of targets). Obviously, the former has fewer states than the latter, and thus the robots using the former controller are more likely to be in the same state. Therefore, the learning coordination algorithm may work better for the former controller (fuzzy reinforcement learning) than the latter (learning with the behavior-based control network).

Videos and photos have been taken for all simulations. Please kindly refer to http://guppy.mpe.nus.edu.sg/~mpeangh/kevin to review them to have a better understanding of the system.

5.4 Summary

In this chapter, the potential field-based target-tracking algorithms are introduced. In addition to the traditional potential field-based approaches, a weight is added to limit the attractive force to the targets. By selectively adjusting this weight, the robots can cooperate to assign targets to a suitable robot for tracking (Liu et al., 2003, 2004a, 2004b). For example, when two robots are tracking the same target, they will compare their distances to the target, and then the more distant

robot will leave to search for other targets. This coordination can be achieved by distributed algorithms without the need for intercommunications.

In the proposed application scenario, the number, location, and mobility patterns of targets are unknown. This increases the uncertainty for system design as it is difficult to determine the important parameters, e.g., weight-decrease ratio, for the hardcoded controllers to achieve optimal target assignment. To address this problem, two reinforcement learning algorithms are proposed to allow the robots to learn cooperative tracking (Liu et al., 2004c, 2005a, 2005b, 2006). In contrast to the hardcoded controllers, the learning controllers do not need the designer to select any important parameters to achieve high performance. The simulations show that the proposed learning controllers have the ability to adapt to different scenarios. Furthermore, in both homogeneous and heterogeneous robot teams, the performance of the learning robots is satisfactory and consistent, thus highlighting the advantages of learning controllers. In addition, to eliminate the interference among concurrently learning robots, a distributed learning coordination algorithm is proposed and applied to both learning controllers. This algorithm is shown to be effective for the avoidance of undesired learning results.

6 MULTI-ROBOT MOBILITY-ENHANCED LOCALIZATION

In the proposed surveillance application, it is important to obtain the location information of the static sensors, mobile robots, and targets. Because the number of sensors and robots is large, it is practical and feasible to apply the hop-count-based localization scheme (Bischoff & Wattenhofer, 2004), as it is scalable and can be easily implemented in ad hoc networks (Frodigh et al., 2000) without adding extra sensors or equipment. However, if the sensors are sparsely distributed in the networks, the location estimation may be quite coarse due to the inaccuracy of the hop-counts in such scenarios.

To achieve satisfactory localization in the proposed application scenario, a multi-robot system is developed to let the mobile robots move to the critical areas (where sensor density is sparse) to increase sensor density, and consequently improve localization accuracy (Sit et al., 2007). In the following parts of this chapter, hop-count-based localization and its limitations are introduced in Section 6.1. Following this, Section 6.2 presents the implementation of the proposed auction-based task allocation algorithm for multi-robot mobility-enhanced localization. Simulation and discussions are presented in Section 6.3. Finally, Section 6.4 concludes this chapter.

6.1 Hop-Count-based Localization

As introduced in Section 3.3, in wireless ad hoc networks, each sensor or robot can only directly "communicate" with its neighbors within one-hop range. If the destination sensor is beyond this range, the communication packets have to be ferried by intermediate sensors through multi-hops. Based on this communication methodology, the hop-count (number of hops) between two sensors

is a metric to represent the distance between them. This is the basis for hop-count (connectivity)-based localization.

To apply hop-count-based localization in ad hoc networks, two assumptions should be satisfied. One is that the positions of some nodes in the network are known, and they serve as the reference/beacon nodes that broadcast their positions throughout the entire network. For 2-D environments, at least 3 references (not in a straight line) are needed; the other is that the sensors, which need to learn their positions, can find the shortest links connecting to at least three reference nodes through one hop or multiple hops. Based on these two assumptions, the sensors, by finding the shortest path (minimal number of hops) to the reference nodes, can estimate the distances to the reference nodes (the beacons deployed at the four corners) based on the hop-count. Since the positions of the reference nodes are known, the sensors can then estimate their positions using triangulation (Langendoen & Reijers, 2003). In theory, the localization error of hop-count-based localization is on the same order of magnitude as the hop distance. For many applications that do not have stringent accuracy requirements, such accuracy is acceptable. Regarding the proposed surveillance scenario, an accuracy of within several meters (equal to one-hop length) is enough for the surveillance tasks (e.g., search and rescue) to locate the target (e.g., victim) in the correct place (e.g., a room).

As compared to other localization methods, the hop-count-based localization only requires hop-count information, which is a by-product of the routing protocols in wireless ad hoc networks, e.g., Ad hoc On-Demand Distance Vector Routing (Perkins & Royer, 1999). Therefore, the acquisition of hop-count information does not normally incur additional communication overhead, and is scalable for sensor networks with a large number of sensors. Examples of hop-count-based localization algorithms are DV-Hop (Niculescu & Nath, 2003) and Hop-TERRAIN (Savarese et al., 2002).

One main drawback of hop-count-based localization is that the accuracy of the location estimation depends largely on the distribution of sensors (density) in network. If the sensor density is low, each sensor has very few neighbors, and the links between a sensor (to be localized) and reference nodes (beacons) may require more hops as compared to dense networks (Niculescu & Nath, 2001). This problem is illustrated in Figure 6-1, where the arrows indicate the shortest link (the connection with lowest number of hops) from sensor *A* to *B*. If the sensor density is high (Figure 6-1-a), sensor *A* can easily find a "straight" path to sensor *B*; however, if the sensor density is low (Figure 6-1-b), the shortest link from sensor *A* to sensor *B* has to be "indirect". Although the physical distance between sensors *A* and *B* are the same in both scenarios, the hop-counts between them are different. Obviously, the overestimated hop-count numbers in sparse networks may lead to large errors in distance estimates.

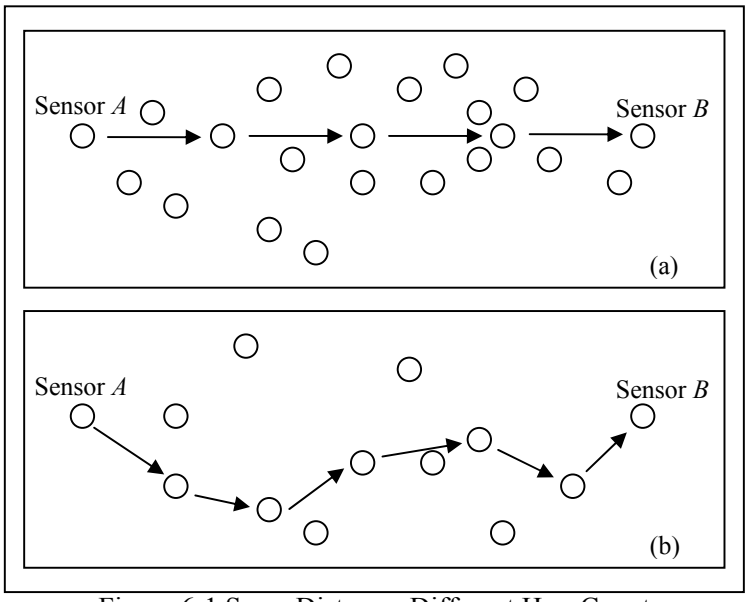

Figure 6-1 Same Distance, Different Hop-Count

To address this problem, two classes of approaches are proposed. One is to adjust the distance estimation according to the sensor density. For example, the DV-Hop algorithm (Niculescu & Nath, 2003) refines the location estimation by adjusting the hop distance according to the hop-

counts between reference nodes; while the Density-aware Hop-count-based Localization (DHL) (Wong et al., 2005) adjusts the hop distance of the sensor according to the number of neighbors. By these means, the hop distance can be reduced in sparse networks where the links between sensors may take extra hops. Consequently, the errors in distance estimation are reduced. However, this class of methods usually cannot improve the topology of the network, e.g., connect nodes with shorter link(s).

Another class of approach to address the accuracy problem in sparse networks is to increase the sensor density of the sparse areas. Lim and Rao (2003) introduced mobile robots to traditional static ad hoc networks. By allowing multiple robots to move randomly within the network, the sparse areas will become denser. The robots can then serve as intermediate sensors to "bridge" the static sensors using shorter paths. This approach is shown through simulations to be an improvement over traditional, static sensor, hop-count-based localization methods. However, with random mobility, the mobile robots usually take an excessively long time to move to the critical locations where sensors are sparsely distributed. Hence, the performance gains from the random mobility of mobile robots are neither reliable nor predictable. In this study, this problem is addressed by enabling the mobile robots to move to the critical areas through cooperation.

6.2 Auction-based Cooperation for Enhancing Localization

To enhance hop-count-based localization, this research aims to enable mobile robots to move to the areas where the sensors are sparsely distributed, thereby increasing the sensor density to improve the localization precision.

Essentially, this is a task allocation problem, in which the movements to specific locations are the tasks to be performed, and the resources are the mobile robots. In general, there are four main research issues, as follows:

- Where are the most desirable locations to move to? – where to move?
- Once these locations have been identified, how should the robots cooperate in order to move to these locations? In other words, which robot(s) should move to the desired area(s)? – who to move?
- Assuming that a robot has been allocated a location to move to, how should the robot then move to that location? – how to move?
- What if the robot fails? How should the algorithm recover from failure? – failure recovery.

6.2.1 Where to Move

In consideration of the main research issues, the first important point is to identify the locations to which the robots are required to move. As outlined in Section 6.1, the accuracy of hop-count-based localization is affected by the sensor density in the network: areas with low sensor densities are more likely to generate localization errors. At this juncture, it is meaningful to introduce an additional assumption: the number of neighbors of a sensor is an indicator of the local sensor density of that area. This is a reasonable assumption, as a sensor with few neighbors within its direct communications range would suggest that the area is sparse with low sensor density. In this study, the robots will move towards the static sensors with few neighbors. To achieve this, an auction-based task allocation scheme is proposed to assign robots to move to the sparse areas. Static sensors with insufficient numbers of neighbors (and hence likely to be in low sensor density regions) will become the auctioneers that issue move requests, retrieve the bid from the

robot(s), and then assign the robot(s) to move. As compared to the MURDOCH algorithm, which is an auction-based task allocation framework "built upon a principled, resource-centric, publish/subscribe communication model" (Gerkey & Mataric, 2002), this approach is unique in the following aspects:

- In this study, the auctioneer to initiate tasks is a static sensor, but not a robot. In the proposed surveillance scenario and system, the number of static sensors is much larger than the number of robots; hence, the static sensors are more likely to find the sensor sparse areas that need robots to approach.
- In this study, the negotiation among robots and static sensors is through ad hoc communications, whereby information is transmitted directly (within one-hop range) or via multi-hops. This communication methodology is scalable for large number of static sensors and mobile robots, comparing to traditional long-range broadcasting sensor networks.
- Based on ad hoc communication methodology, it is easy to limit the communications to a sub-region of the environment by setting the maximal number of communication hops. This can reduce the communication overhead without degrading the system performance, because in many cases only the robots near the task area should be involved.

In the environment, there may be a lot of static sensors having very few neighbors, resulting in multiple sparse areas. Therefore, they will all require the robots to approach them. In this approach, there is only one active auctioneer at any one time. Hence, there is a need to select a static sensor to be the auctioneer. Since the sensor with fewest neighbors is likely to have the largest localization estimation error, it is given the priority to become the auctioneer.

In the proposed surveillance system, each static sensor may periodically send a one-hop broadcast message to its neighbors. This message can help each sensor learn the number of neighbors that it

has. If a sensor has very few neighbors (i.e., less than the "*Least_Neighbor_Threshold*"), it will flood this information throughout the entire network. The sensors will then be able to determine the sensor which has the least number of neighbors. That particular sensor will become the auctioneer. The details of this distributed algorithm are shown in Algorithm 6.1. Each static sensor executes this algorithm.

Algorithm 6.1 Selection of Auctioneer. This algorithm is executed for each static sensor i.

Step 1. Initialize all values: Let N_i (number of neighbors of sensor i) = 0, L (ID of the static sensor with least number of neighbors) = 0, N_L (number of neighbors of sensor L) = 0.

Step 2. Broadcast <*i, "hello"*> to one-hop neighbors.

Step 3. Listen for a finite period of time (length of time should be in multiples of the actual communications time).

If sensor i receives a broadcast from any of its one-hop neighbor during this time, e.g., <*j, "hello"*>, let $N_i = N_i + 1$.

Step 4. Check the value of N_i.

If $N_i >=$ *Least_Neighbor_Threshold* (e.g., 6), wait until a trigger signal is received before going to Step 1. (Waiting means i is in a dense area and there is no need to participate in the auction.)

Step 5. Let $L = i$; $N_L = N_i$.

Step 6. Broadcast <i, N_i, *"neighbor info"*> throughout the entire network.

Step 7. Listen for a finite period of time (length of time should be in multiples of the actual communications time).

Each time sensor i receives a message, e.g., <m, N_m, *"neighbor info"*>,

If $N_m < N_L$, let $L = m$; $N_L = N_m$ (choose the sensor with the least number of neighbors).

If $N_m = N_L$ and $m < L$, let $L = m$; $N_L = N_m$ (if two sensors have same number of neighbors, choose the sensor with the lower ID).

Step 8. If $L = i$, then sensor i is assigned as the auctioneer, and the auction begins.

Else, wait until a trigger signal is received before going to Step 1.

In this distributed auctioneer selection algorithm, the static sensors have to be coordinated for the purpose of broadcasting and waiting. A sensor is randomly selected as a coordinator during the start. (For example, it can be sensor 0.) The coordinator begins the selection process according to Algorithm 6.1 and sends the required triggers (for Steps 4 and 8). When a sensor is selected as the auctioneer, it will request the robots to approach it. This sensor (auctioneer) will also serve as the coordinator to select the next auctioneer.

Since the static sensor does not have absolute positioning sensors or equipment, the location estimate of this sensor is obtained by hop-count-based localization. For a 2D environment, a sensor can only localize itself when it is connected to no fewer than 3 beacons for triangulation. Therefore, a sensor will only run the selection process (Algorithm 6.1) if its *k-Conn* (i.e., *k*-connectivity introduced in Section 4.3) is 3 or greater. As introduced in Section 4.3.2.2, in the proposed surveillance scenario, the robots could improve the connectivity of the network and thus the average of *k-Conn* could be larger than 3. This means all or nearly all static sensors can locate themselves, and they can participate in the selection of the auctioneer.

6.2.2 Who to Move

After acquiring knowledge on where the robots should move to, the next issue is to select the most suitable robot(s) to move towards the area. In the proposed auction scheme, the auctioneer will broadcast the task (indicated by its own position) throughout the entire network. The robot

that receives this message will provide a bid (indicating the robot's position and speed) if it is free (i.e., the robot is not executing other tasks). In order to minimize the time that the robot(s) spend in moving to the locations, the auctioneer will choose the most suitable robot(s) to do the task by awarding a contract. For this purpose, a metric which takes into account both position and speed is used to select the most suitable robot(s): the nearest robot(s) with the highest speed is (are) awarded the contract (i.e., assigned the task to move towards the auctioneer). Details of the auction are presented in Algorithm 6.2 (from the perspective of the auctioneer) and Algorithm 6.3 (from the perspective of the robot). It should be noted that in the auction, since the static sensors and mobile robots do not have absolute positioning sensors, their location information is estimated by hop-count-based localization.

Algorithm 6.2 Auction from perspective of Auctioneer (for the selected static sensor which is the auctioneer, say i)

Step 1. Estimate own location $<x_i, y_i>$ by hop-count-based localization.

Step 2. Set *RobotPool* (number of robots already recruited) = 0; R (number of robots needed) = $Least\_Neighbor\_Threshold - N_i$. ($N_i$ is the number of neighbors)

Step 3. Broadcast $<i, x_i, y_i,$ *"auction request"*$>$ to all robots.

Step 4. Listen for a finite period of time.

If a reply is received from robot r during this time, e.g., $<r, B_r>$ (r is the ID of robot, B_r is the bid offered by robot r), record this information in a queue Q, in which the robots are ordered by the value of their bids. (The highest bid is placed at the beginning of the queue), and let $RobotPool = RobotPool + 1$.

Step 5. Compare R and *RobotPool*.

If $R > RobotPool,$ the available robots are not sufficient,

Unicast a contract $<i, x_i, y_i,$ *"contract"*$>$ to each of the robots in Q, and

Let $R = R - RobotPool$, and Goto Step 3.

Else,

Unicast contracts <*i*, x_i, y_i , *"contract"*> to each of the first *R* robots in *Q*

Step 6. Finish auction.

(Note that B_r is calculated by the robot offering the bid using Algorithm 6.3.)

Algorithm 6.3 Auction from the perspective of the Robots (e.g., robot *r*)

Step 1. Estimate the location <x_r, y_r> by hop-count-based localization.

Step 2. Listen for a finite period of time.

If robot *r* receives an auction request from the auctioneer, e.g., <*i*, x_i, y_i, *"auction request"*>, send a bid as <*r*, B_r>. B_r = speed(*r*)/dist (<x_i, y_i>, <x_r, y_r>). The *Br* is the inverse of the time needed to reach target location.

Step 3. Listen for a finite period of time.

If a contract is received from the auctioneer, e.g., <*i*, x_i, y_i , *"contract"*>, set status as busy. Leave this process.

Else goto Step 1.

Since the robot does not have absolute positioning sensors or equipment, the location estimate of this sensor is obtained by hop-count-based localization. Therefore, a robot will only participate in the auction process (Algorithm 6.3) if its *k-Conn* is no less than 3 and it is free (i.e., not working on other tasks).

When the auctioneer has assigned the task to the selected robot(s), it will monitor its neighborhood continuously. If the required robots have arrived and the density of the neighborhood (number of neighbors) reaches the *Least_Neighbor_Threshold*, the auctioneer will release the recruited robots by canceling the contracts. The static sensor will not "call for help"

(hold new auctions) in the future. The freed robots can participate in the next auction or other tasks. It should be noted that a robot that is working on a task will not participate in new auctions until it has been freed.

6.2.3 How to Move

After a robot has been assigned a target location, it needs to move towards that location with obstacle avoidance based on its local sensing and communications. One simple solution is to use the Artificial Potential Field (APF) method. In this approach, the location of the target sensor (auctioneer) becomes the attractive force source for the robot when it is assigned a task. It should be noted that both the target location and robot location are estimates based on the localization scheme.

As in the potential field-based control introduced in previous chapters, the robots may be trapped in local minima where the attractive forces and repulsive forces balance each other. In the simulation scenario to test the multi-robot mobility-enhanced localization, there are no large obstacles in the environments; therefore, the local minima are less likely to appear. In addition, because both the target location and robot location are estimates and may change, the orientation of the attractive force is not a fixed value. Therefore, the robot is less likely to be trapped because the attractive forces are not quite stable.

6.2.4 Failure Recovery

The last main research issue concerns fault tolerance. Due to the assumption that the robots may fail at any time, as well as the possibility that the robots may fail to find a path to the goal due to

obstacles along the way or the local minimum problem inherent in APF methods (Koren & Borenstein, 1991), there is a need to consider task failure from the perspective of the robot. To address this problem, contract renewal is used, whereby the assigned task expires within a preset time interval. If a robot cannot reach the target sensor within specified time interval, or loses communications with the auctioneer of the task, it is deemed to have failed its task. The robot will give up its current task and participate in the next auction. From the perspective of the auctioneer, the auctioneer that issues the move request will then initiate another auction to assign the work to other robot(s) to replace the failed robot(s).

A further consideration is that the static sensors may also fail at any point in time. This may, for example, be in the form of communication failures. As a result, the robots which have been allocated tasks by the auctioneer (the static sensor with communication failures) may not be freed from the tasks if the static sensor fails to free it from the contract. This can be dealt with using a similar implementation of a "time out" for the mobile robots. If the mobile robot has not been freed from the task after a sufficiently long time period, the robot will assume static sensor failure, and then free itself from the task so that it can participate in the next auction.

6.2.5 Localization

Both static sensors and mobile robots need to estimate their positions to enable the proposed intelligent mobility. In auction-based task allocation, the auctioneer needs to broadcast its position to the robots; on the other hand, the robots need to provide bids containing their location information.

In this study, the sensor or robot can obtain rough estimates of its distance to the beacons by hop-counts. Since the locations of beacon nodes are known, the position of a sensor or robot can then be calculated by triangulation.

6.3 Simulation Tests and Discussions

6.3.1 Simulation Environment and Settings

To measure the efficacy of the proposed "intelligent" motion approach, it may be benchmarked against the "random" motion approach (Lim & Rao, 2003). The simulation environment and settings are as follows:

- The simulator: Webots (Olivier, 2004).
- Simulation environment: 45 x 45m square area including 4 beacons (reference nodes), 30 static sensors and 10 mobile robots. The robots, beacons, and static sensors are represented by circles with radius less than 1m.
- In the environment, the reference nodes are placed at the 4 corners, and the static sensors are randomly placed. The initial positions of the robots are also random.
- The communication range of each sensor/robot is 8m.
- The update interval of the entire network by wireless communications is 1.6 seconds.
- The performance results are obtained from the average of 50 runs. Each simulation run is 10000 steps long. Each simulation step is about 0.16 second in the real world.
- In the "intelligent" motion approach, varying values of *Least_Neighbor_Thresholds* of "denseness" are tested. For example, when the threshold is 6, a static sensor with 2 neighbors will require 4 robots to approach; however, if the static sensor has 6 neighbors,

it will not require any robots to approach it. The values of *Least_Neighbor_Thresholds* that are tested are 4, 6 and 8 for the 30 static sensors.

6.3.2 Simulation Results and Discussion

With respect to hop-count-based localization, the error in distance estimation will affect the resultant location estimation during the triangulation process, but the trends of these two kinds of errors are not necessarily consistent. Therefore, it is better to use both distance and location estimation errors to evaluate the performance of the localization algorithms. Since the robots are constantly moving and their real locations are always changing, in this study only the estimation errors of static sensors are presented and discussed.

For each static sensor, the distance error is the average of the differences (absolute values) between the estimated distance and real distance to the four reference nodes; the location error is the distance between the estimated location and real location of the sensor. Figure 6-2 shows the average distance and location estimation errors at the end of the simulation. The intelligent mobility (with *Least_Neighbor_Thresholds* = 4, 6) achieves better localization performance than the random mobility. Through auctions, the robots can proactively and intelligently move towards the most critical areas (which have sparse sensor distributions) to serve as "bridges" to shorten the link between static sensors and reference nodes.

Figure 6-3 and Figure 6-4 show the decrease in the distance/location estimation errors during the simulation (with *Least_Neighbor_Thresholds* = 6). The intelligent mobility can reduce the errors faster than the random mobility. By observation of the behavior of the robots in the simulation, we also see that when the robots are in intelligent mobility mode, several robots that have been assigned the same target sensor to approach will usually move in a group. This motion pattern

further increases the possibility of constructing "bridges" between static sensors and reference nodes.

It is also found that different thresholds of "denseness" yield different performance. As shown in Figure 6-2, a *Least_Neighbor_Thresholds* of 4 generates the best result. With increasing threshold values, the performance of the system deteriorates. When the threshold reaches a value of 8, the intelligent mobility provides even worse performance than random mobility. A possible explanation for this phenomenon is that if the *Least_Neighbor_Thresholds* is too large, the static sensors will require more robots to approach. This may decelerate the progress because more time is required to wait for the last robot to arrive before the target static sensor can release the robots to work on the next task. Furthermore, when the number of neighbors is above a threshold, e.g., the magic number of 6 in this case (Kleinrock & Silvester, 1978), increasing the neighbor number may not greatly improve the localization precision.

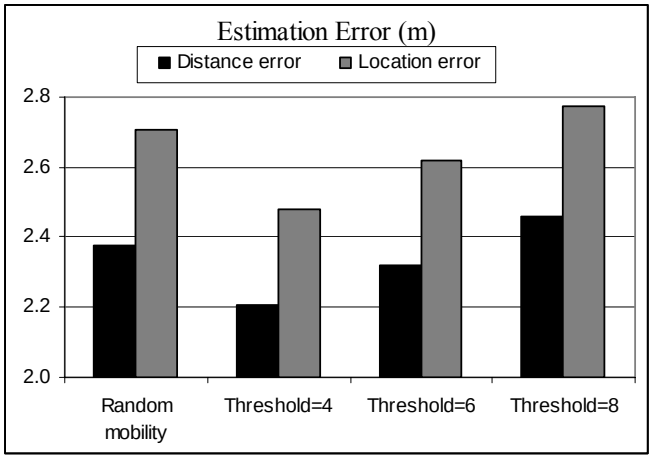

Figure 6-2 Distance and Location Estimation Errors (At the end of simulation)

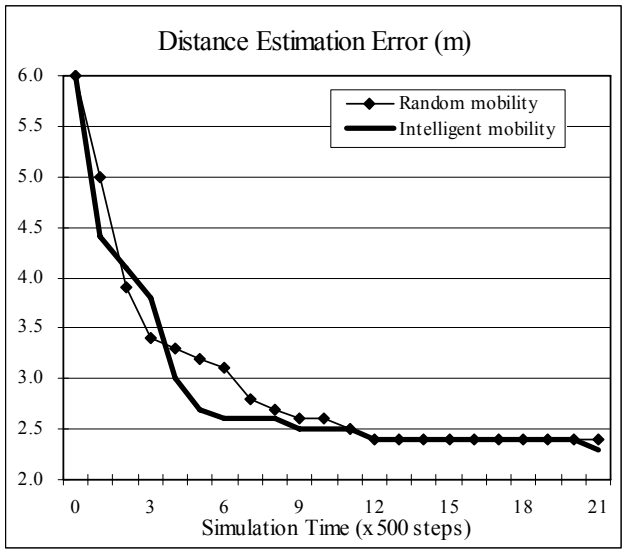

Figure 6-3 Distance Estimation Error (For the intelligent mobility, threshold=6)

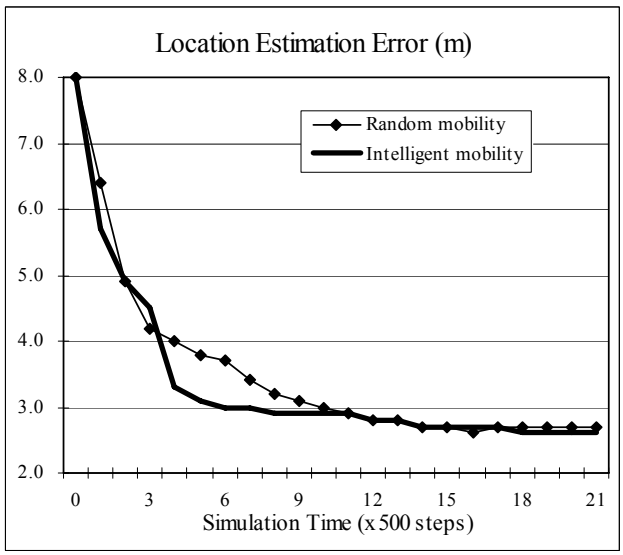

Figure 6-4 Location Estimation Error (For the intelligent mobility, threshold=6)

Another significant difference between the random and intelligent mobility is in the performance fluctuations. This can be seen in Figure 6-5 and Figure 6-6, which show the distance and localization error for each approach after 10000 simulation steps in each simulation run. Random mobility produces much more variations in performance than intelligent mobility across 50 runs of the simulation. These significant fluctuations in performance between each run for random mobility are evident in both the distance error and localization error. The standard deviation of

the distance/location estimation errors of the 50 runs in the last simulation step is shown in Figure 6-7. The intelligent mobility mode can achieve more repeatable and consistent estimations. The lack of repeatability and consistency for random mobility can be attributed to the fact that the mobile robots move about in random, unpredictable directions; hence, the improvement in localization information depends largely on the probability that the robots move to locations with low sensor densities. In contrast, the performance gains through the use of intelligent mobility are much more repeatable and consistent, due to the fact that the mobile robots know the approximate locations of the areas with low sensor densities; hence, moving towards these locations will help to improve the localization accuracy.

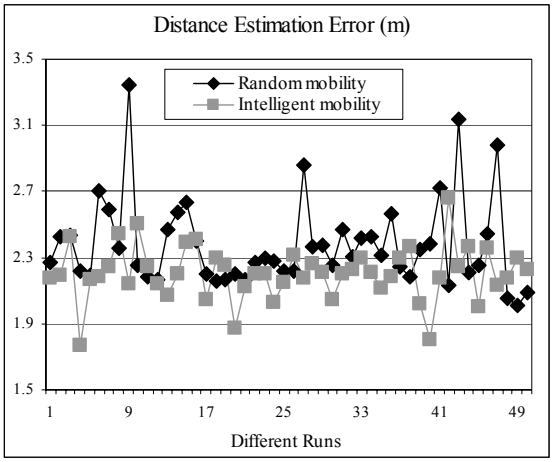

Figure 6-5 Distance Estimation Error (For the intelligent mobility, threshold=4)

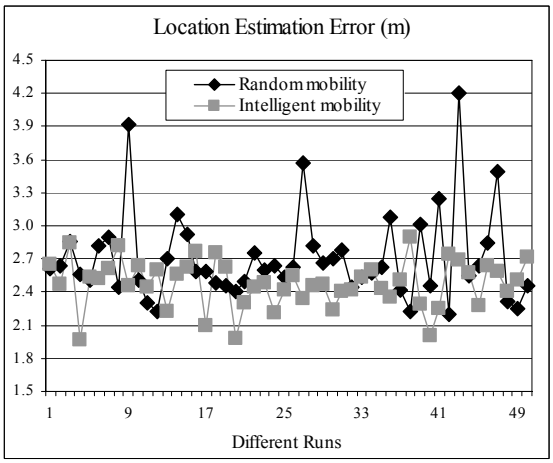

Figure 6-6 Location Estimation Error (For the intelligent mobility, threshold=4)

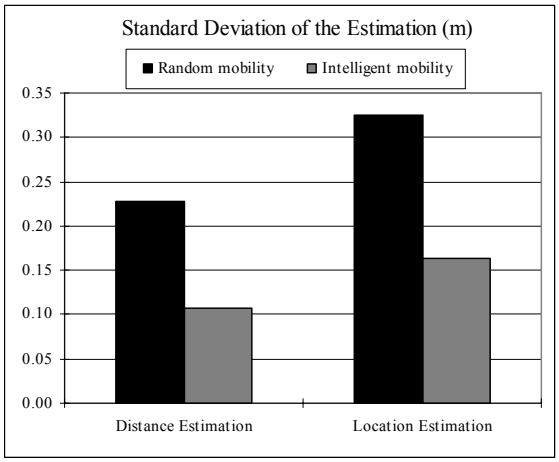

Figure 6-7 Standard Deviation of the Estimation Error (For the intelligent mobility, threshold=4)

Videos and photos have been taken for all simulations. Please kindly refer to http://guppy.mpe.nus.edu.sg/~mpeangh/kevin to review them to have a better understanding of the system.

6.4 Summary

In this chapter, an intelligent multi-robot approach is proposed to improve the accuracy of hop-count-based localization in the proposed surveillance scenario (Sit et al., 2007). This algorithm is developed with consideration of efficiency and fault tolerance. The core of this algorithm lies in the auction-based task allocation algorithm without a fixed auctioneer, such that any sensor in the network can "call for help" through self-discovery of its surrounding sensor density.

Through simulations, it is shown that this new intelligent mobility model out-performs the random mobility model, both in terms of distance error and location error. The intelligent mobility model is also found to produce more consistent and repeatable results as compared to the random mobility model.

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this thesis, a series of distributed multi-robot surveillance approaches are presented. As compared to traditional surveillance systems, this study emphasizes "practicality" in that realistic assumptions are made with respect to the environmental conditions, robot capabilities and sensor limitations. In addition, this study also highlights the advantages of cooperation in multi-robot systems. Distributed and scalable intelligent control methodologies are developed for efficient cooperation among the robots in multi-robot systems.

7.1.1 Practical Surveillance

In this thesis, a multi-robot surveillance system has been designed to self-organize together with static sensors to construct a hybrid sensor network. This system can explore and monitor unknown environments, search for targets, track targets, and provide useful location information.

To explore the environment and find the targets (mobile/static, embodied/virtual), both exploration and target searching problems are developed (Seah et al., 2006, 2006). Three exploration algorithms, (i) potential field-based exploration; (ii) swarm intelligence exploration; and (iii) landmark-based exploration, are proposed to increase the coverage to aid robots in target searching. In addition, a searching algorithm, hop-count gradient-orientated searching, is introduced to enable robots to search for targets in promising areas by following useful clues. Realistic simulation integrating robotics and communication simulators demonstrate the efficacy of these algorithms. In addition, potential field-based exploration and swarm intelligence

exploration are implemented with real robots, in both small and extended environments. These experiment tests further highlight the efficacy of the algorithms.

After the mobile targets have been found, it is important to track them for continuous observation. An advanced artificial potential field-based intelligent tracking algorithm has been proposed for this purpose (Liu et al., 2003, 2004a, 2004b). This algorithm enables robots to "select" suitable targets to track (target assignment) according to their capabilities and feasibilities. As compared to other artificial potential field-based approaches, this algorithm is more intelligent in that a robot can compare its situation with other robots and then make appropriate decisions in a distributed manner. The algorithm is highly scalable; thus it can be applied in large robot teams. Since the targets are mobile and their mobility patterns are usually unknown, cooperative tracking is more difficult than exploration and target searching. To increase the adaptivity of robots and to avoid hardcoded designs, two reinforcement learning-based approaches, reinforcement learning in a behavior-based control architecture (Liu et al., 2004c, 2005b, 2006) and fuzzy reinforcement learning (Liu et al., 2005a), are proposed. These learning algorithms enable the robots to learn cooperation based on robot-robot and robot-environment interactions. In addition, a simple and distributed learning coordination scheme is developed to allow the robots to learn concurrently with minimal interference among them. Simulation tests show that the proposed algorithms work well in the proposed surveillance environment.

With respect to the localization issue, a simple and scalable localization method known as hop-count-based localization is introduced (Sit et al., 2007). To address the intrinsic problem of hop-count-based localization, an auction-based task allocation scheme is proposed to enable multiple robots to improve the localization accuracy. Using only a few robots, the localization accuracy can be remarkably improved. The proposed intelligent mobility model is shown to be more

effective than the random mobility model by simulation tests. The improvement of the localization accuracy is also better and more consistent.

In this thesis, the proposed multi-robot surveillance system covers most aspects of typical surveillance problems. The exploration and searching algorithms allow the robots to find the targets easily. When the mobile targets are found, the robots can track them using tracking algorithms. If the environment is non-stationary or the scenario is complex, the learning capabilities of the robots enable them to choose optimal strategies for tracking. In addition, the localization methods can provide accurate location information of targets and robots. The proposed algorithms are simple and scalable; they could be implemented in most real-world surveillance applications. Both simulation tests and real hardware tests have been conducted to demonstrate the efficacy of these algorithms.

7.1.2 Distributed Cooperation Methodology

In multi-robot surveillance systems, it is important to achieve cooperation among robots so that they can achieve better performance as compared to single-robot system or less cooperative systems. In this thesis, several distributed cooperation algorithms have been proposed. These algorithms are applicable to most real applications.

In exploration and target searching, the robots can effectively utilize communications to achieve cooperation. The robots can share their heading information to move along different directions (swarm intelligence exploration); this can help to disseminate the robots into different regions of the environment to improve the coverage. The robots are also able to cooperate with static sensors (landmark nodes) by communications; the robots are then able to move to different regions and avoid re-exploration of the covered regions (landmark-based exploration). Furthermore, with

respect to the proposed hybrid sensor network that includes both mobile robots and static sensors, the cooperation between robots and static sensors can help the robots to find the communication gaps (virtual targets) efficiently (hop-count gradient-oriented target searching). This is a self-organizing and self-improving enhancement of traditional sensor networks which consist of static sensors only.

In tracking, due to the movement of targets in an unknown mobility pattern, the robots have to reactively find the optimal tracking strategy according to their observations. In this thesis, the robots make use of indirect communications to track targets in a cooperative manner. Even when no messages are exchanged among robots, the robots can still achieve high levels of cooperation in target assignment through the selective-adjust heuristic of the artificial potential field-based tracking. In addition, two learning algorithms are proposed to further increase the adaptivity of the system. As the learning algorithms are totally distributed, the learning performance may be degraded if the robots interfere with one another. To solve this problem, a distributed learning coordination algorithm, which is both efficient and scalable, is developed to eliminate the interference among robots.

Hop-count-based localization is suitable for the proposed surveillance scenario because it is simple and scalable for large hybrid sensor networks including both mobile robots and static sensors. In this thesis, the robots cooperate with static sensors to identify critical regions and assign the most suitable robots to approach such regions. This is accomplished by an auction-based task-assignment algorithm. As compared to traditional auction-based task-allocation algorithms, the proposed algorithm does not require a fixed auctioneer: each static sensor may be elected as the auctioneer and hold the auction. The system is more robust and scalable.

In this thesis, different cooperation methodologies are used for different surveillance tasks. They can achieve the desired high level of cooperation using little or no direct communications. The efficacy of these algorithms is shown by both simulation and real tests.

7.2 Future Work

Surveillance is a wide research topic covering many areas such as exploration, target searching, target tracking, localization, etc. In this thesis, a series of multi-robot systems are proposed for surveillance tasks. These algorithms can extend the coverage of the environment, accelerate the searching of targets, enhance the performance of target tracking and monitoring, and increase the reliability and robustness of the entire surveillance system.

To improve multi-robot cooperative surveillance, some important research issues that should be further studied are highlighted in this subsection.

7.2.1 Surveillance

7.2.1.1 Exploration and Target Searching

In multi-robot cooperative surveillance, the aim is to achieve satisfactory observation of the environment or objects of interest (e.g., targets). Exploration and targets searching are critical for successful surveillance. In this thesis, some algorithms have been proposed to conduct exploration and target searching without the need for map building. This simplifies the algorithms and lowers the requirements of the computation ability, memory storage and sensor capability of the robots. However, as the robots do not have the map of the environment, and do not build a

map during the exploration process, it is difficult to ensure that the robots can remember the covered regions and avoid re-exploration of such regions. Although the landmark-based exploration algorithms can utilize some static sensors to remember exploration history and instruct the robots to move towards uncovered areas, the exploration performance is largely dependent on the deployments of landmarks. If there are insufficient landmarks (i.e., the landmark sensors cannot cover the whole area), the robots cannot guarantee the full coverage of the environment.

To solve this problem, intuitively the robots should have some kind of map to help them in exploration and target searching. This is known as the Simultaneous Localization and Mapping (SLAM) problem. However, as introduced in Section 2.1.1.1, SLAM has some disadvantages in its implementation.

Based on the proposed surveillance system, which includes both intelligent mobile robots and static sensors, it will be quite interesting to study if traditional SLAM algorithms can be revised and modified to fit this surveillance system. The following are some possible directions for such improvements:

- Better map representation. In the proposed surveillance system, the static sensors can not move. They can serve as landmark nodes to guide the robots in better exploration. If these landmark nodes can share and merge their information, they can construct node-based maps. This map is different from the occupancy map, feature map, or topology map; however, this map can be quite useful during robot exploration. In addition, the building of the node-based map only requires intercommunications among static sensors. This is achievable by most ad hoc sensor networks.
- Better path planning. In this thesis, there has not been much discussion of the path planning problem because the proposed algorithms are reactive and real-time. However,

when the robots have obtained some information of the environment, such as the hop-count information, they may utilize such information to plan the routines for better exploration. For example, the robots may try to move from one landmark node to another so that they can cover more areas.

- Better cooperation for observation. In multi-robot systems, it is possible that the robots are specialized in that they are equipped with different types of sensors. It is non-trivial to coordinate such robots to achieve better observations. For example, to detect and monitor a fire in the environment, the robot(s) with heat sensors have to cooperate with the robot(s) with video cameras.

7.2.1.2 Target Tracking

In a surveillance system, it is necessary for the robots to have the ability to track mobile targets for continuous and close observations of the targets. In this thesis, artificial potential field-based approaches are proposed to enable target assignment among robots. These approaches mainly focus on the cooperation among robots and there is little emphasis on the environmental constraints in tracking.

In a complex environment with walls and doors, target tracking is more difficult than in an open area because the targets can hide behind the walls. To avoid the loss of targets, it is crucial that the robots consider environmental factors. Some possible solutions for this problem are as follows:

- Estimation of the probability to lose targets. The robots should detect the environment and find the possible objects (walls or doors) that the targets may hide behind. If there is

more than one object, the robot should be able to evaluate and rank the level of danger of each object.

- Estimation of the mobility of the targets. If the robots can observe and predict the motion of the targets, they can have better tracking performance.

7.2.1.3 Localization

In surveillance systems, the location information of the robots, sensors and targets is essential because without this information, the sensor data may not be meaningful. In this thesis, intelligent robot mobility is introduced to improve the hop-count-based localization. To further improve the localization accuracy, the following work can be carried out:

- Data fusion. In the proposed surveillance system, only the hop-count information is used for localization. It can be quite helpful if the robots can input their sensor data to improve localization. For example, the robots may use odometry to estimate their movements. It will be quite interesting to study how odometry data can be fused with hop-count information to obtain better location estimates.
- Other localization methods. In addition to hop-count-based localization, other methods may be suitable for the proposed surveillance system. For example, TDOA (time difference of arrival) methods have accurate distance estimates and can improve the localization accuracy.

7.2.2 Cooperation

7.2.2.1 Control Methodology

In this thesis, different cooperation algorithms are proposed for the surveillance tasks. Such algorithms are distributed and thus scalable for a large number of robots. To further improve the performance of surveillance, the following studies are significant:

- Specialization. In real applications, it is not practical to equip each robot with the same sensors, especially since the sensors can be quite expensive. In this case, the robots may have different capabilities. An important research topic is the effective coordination of a heterogeneous robot team and the optimal utilization of the robot resource.
- Robot-sensor cooperation. In the proposed surveillance system, there are both mobile robots and static sensors. It will be quite interesting if the robots and sensors are able to achieve high levels of cooperation to improve the surveillance.

7.2.2.2 Robot Learning

Multi-robot concurrent learning on cooperation is one of the ultimate goals of robotics and artificial intelligence research. In this thesis, two reinforcement learning-based learning algorithms are proposed to enable the robots to generate cooperative behaviors in continuous space. In addition, a nature-inspired distributed learning control algorithm is developed to coordinate the concurrent learning processes. This algorithm can help to avoid the generation of local sub-optimal control policy or the cyclic switching of control policies without the need for explicit intercommunications among robots.

In the learning controller that integrates reinforcement learning with a behavior-based control network, the reinforcement learning module has to retrieve discrete input states (target/robot number) and perform discrete actions (weights). A more challenging task is to design a totally continuous and infinite space learning algorithm, and enable the robot to perform state/action discretization through learning. This is an important research issue to be studied. Another problem associated with the learning controller is that the proposed behavior-based control network is specific to the tracking task. If other tasks are selected, e.g., cooperative table carrying, the specific behavior-based control network has to be re-designed accordingly. If the network is inappropriately designed and does not fit the requirements of the task, the reinforcement learning may not work optimally, e.g., it may generate fatal errors of local sub-optimal control policy. Therefore, the performance of the system can be greatly improved if the behavior-based control network in our learning controller is generic and effective for all types of control problems. This is another important research issue to be studied.

In the fuzzy reinforcement learning controller, the fuzzy states and actions are defined by the designer and are specific to the tasks and applications. Like the learning controller that is integrated with the behavior-based controller network, if other tasks are selected, the specific fuzzy states and actions have to be re-designed accordingly. It would be much more efficient if the fuzzification of the normal states and actions can be generic and effective for all types of control problems. This is an important research issue to be studied.

In both distributed learning controllers, the interference among concurrently learning robots may cause the distributed learning controllers to generate unsatisfying results. A critical future research topic is to examine how concurrent learning processes can be perfectly coordinated using minimal intercommunications.

BIBLIOGRAPHY

Arai, Y., T. Fujii, H. Asama, Y. Kataoka, H. Kaetsu, A. Matsumoto and I. Endo. Adaptive Behavior Acquisition of Collision Avoidance among Multiple Autonomous Mobile Robots. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, September 1997, Grenoble, France, pp.1762-1767.

Asada, M., E. Uchibe and K. Hosoda. Cooperative Behavior Acquisition for Mobile Robots in Dynamically Changing Real Worlds via Vision-based Reinforcement Learning and Development. Artificial Intelligence, *110*, pp.275-292. 1999.

Balch, T. and M. Hybinette. Social Potentials for Scalable Multi-Robot Formations. In Proc. IEEE International Conference on Robotics and Automation, April 2000, San Francisco, USA, pp.73-80.

Baronov, D. and J. Baillieul. Reactive Exploration through Following Isolines in a Potential Field. In Proc. American Control Conference, July 2007, New York City, USA, pp.2141-2146.

Bandyopadhyay, T., Z. Liu, M.H. Ang and W.K.G. Seah. Visibility-based Exploration in Unknown Environment Containing Polygonal Obstacles. In Proc. 12th International Conference on Advanced Robotics, July 2005, Seattle, USA, pp.484-491.

Batalin, M.A. and G.S. Sukhatme. The Design and Analysis of an Efficient Local Algorithm for Coverage and Exploration based on Sensor Network Deployment. IEEE Transactions on Robotics *23(4)*, pp.661-675, August 2007.

Bauer, R. and W.D. Rencken. Sonar Feature based Exploration. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, August 1995, Pittsburgh, USA, pp.148-153.

Bischoff, R. and R. Wattenhofer. Analyzing Connectivity-based Multi-Hop Ad-Hoc Positioning. In Proc. 2nd Annual IEEE International Conference on Pervasive Computing and Communications, March 2004, Florida, USA, pp.165-176.

Bjorklund, A. Cooperating Soccer Playing Robots. Master Thesis, Department of Computing Science, Umea University. 2002.

Botelho, S. and R. Alami. M+: A Scheme for Multi-Robot Cooperation through Negotiated Task Allocation and Achievement. In Proc. IEEE International Conference on Robotics and Automation, May 1999, Detroit, USA, pp.1234-1239.

Boyan, J.A. and A.W. Moore. Generalization in Reinforcement Learning: Safely Approximating the Value Function. In Advances in Neural Information Processing Systems 7, ed by G. Tesauro, D.S. Touretzky and T.K. Lee, pp.369-376. Cambridge, MA: MIT Press. 1995.

Burgard, W., M. Moors, D. Fox, R. Simmons and S. Thrun. Collaborative Multi-Robot Exploration. In Proc. IEEE International Conference on Robotics and Automation, April 2000, San Francisco, USA, pp.476-481.

Burgard, W., M. Moors, C. Stachniss and F.E. Schneider. Coordinated Multi-Robot Exploration. IEEE Transactions on Robotics, *21(3)*, pp.376–386. 2005.

Burguera, A., Y. Gonzalez and G. Oliver. Probabilistic Sonar Scan Matching for Robust Localization. In Proc IEEE International Conference on Robotics and Automation, April 2007, Roma, Italy, pp.3154-3160.

Cai, A., T. Fukuda and F. Arai. Information Sharing among Multiple Robots for Cooperation in Cellular Robotic System. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, September 1997, Grenoble, France, pp.1768-1774.

Calisi, D., A. Farinelli, L. Iocchi and D. Nardi Autonomous Navigation and Exploration in a Rescue Environment. In Proc. 2nd European Conference on Mobile Robotics, September 2005, Ancona, Italy, pp.110-115.

Canny, J. The Complexity of Robot Motion Planning. Cambridge, MA: MIT Press. 1988.

Cao, Y. U., A.S. Fukunaga, A.B. Kahng and F. Meng. Cooperative Mobile Robotics: Antecedents and Directions. Autonomous Robots, *4(1)*, pp.7-27. 1997.

Cheng, C.K. and G. Leng. Cooperative Search Algorithms for Distributed Autonomous Robots. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, September 2004, Sendai, Japan, pp.394-399.

Chu, H.T. and B.R. Hong. Cooperative Behavior Acquisition in Multi Robots Environment by Reinforcement Learning Based on Action Selection Level. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2000, Takamatsu, Japan, pp.1397-1402.

Collins, R.T., A.J. Lipton, H. Fujiyoshi and T. Kanade. Algorithms for Cooperative Multisensor Surveillance. Proc. IEEE, *89(10)*, pp.1456-1477. 2001.

Coue, C. and P. Bessiere. Chasing an Elusive Target with a Mobile Robot. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2001, Hawaii, USA, pp.1370-1375.

Doya, K. Temporal Difference Learning in Continuous Time and Space. In Advance in Neural Information Processing Systems 8, ed by D.S. Touretzky, M.C. Mozer and M.E. Hasselmo, pp.1073-1079. Cambridge, MA: MIT Press. 1996.

Dissanayake, G., P. Newman, H.F. Durrant-Whyte, S. Clark and M. Csobra. An Experimental and Theoretical Investigation into Simultaneous Localisation and Map Building (SLAM). In Experimental Robotics VI, ed by P. Corke and J. Trevelyan, pp.265-274. London: Springer-Verlag. 2000.

Efrat, A., H.H. Gonzalez-Banos, S.G. Kobourov and L. Palaniappan. Optimal Strategies to Track and Capture a Predictable Target. In Proc. IEEE International Conference on Robotics and Automation, May 2003, Taipei, Taiwan, pp.3789-3796.

Emery, R., K. Sikorski and T. Balch. Protocols for Collaboration, Coordination and Dynamic Role Assignment in a Robot Team. In Proc. IEEE International Conference on Robotics and Automation, May 2002, Washington, USA, pp.3008-3015.

Fang, G., G. Dissanayake and H. Lau. A Behaviour-Based Optimisation Strategy for Multi-Robot Exploration. In Proc. IEEE Conference on Robotics, Automation and Mechatronics, December 2004, Singapore, pp.875-879.

Faria, G. and R.A.F. Romero. Incorporating Fuzzy Logic to Reinforcement Learning. In Proc. 9th IEEE International Conference on Fuzzy Systems, May 2000, San Antonio, USA, pp.847-852.

Fredslund, J. and M.J. Mataric. A General Algorithm for Robot Formations Using Local Sensing and Minimal Communication. IEEE Transactions on Robotics and Automation, *18(5)*, pp. 837-846. 2002.

Frodigh, M., P. Johansson and P. Larsson. Wireless Ad Hoc Networking: the Art of Networking without a Network. Ericsson Review, *4*, pp.248-263. 2000.

Fujii, T., H. Asama, T. Fujita, Y. Asakawa, H. Kaetsu, A. Matsumoto and I. Endo. Knowledge Sharing among Multiple Autonomous Mobile Robots through Indirect Communication using Intelligent Data Carriers. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, November 1996, Osaka, Japan, pp.1466-1471.

Gerkey, B.P. and M.J. Mataric. Sold!: Auction Methods for Multirobot Coordination. IEEE Transactions on Robotics and Automation, *18(5)*, pp.758-768. 2002.

Gerkey, B.P., R.T. Vaughan and A. Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In Proc. 11th International Conference on Advanced Robotics, June 2003, Coimbra, Portugal, pp.317-323.

Gonzalez-Banos, H.H., C.Y. Lee and J.C. Latombe. Real-Time Combinatorial Tracking of a Target Moving Unpredictably among Obstacles. In Proc. IEEE International Conference on Robotics and Automation, May 2002, Washington, USA, pp.1683-1690.

Grabowski, R. and P. Khosla. Localization Techniques for a Team of Small Robots. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2001, Hawaii, USA, pp.1067-1072.

Grabowski, R., P. Khosla and H. Choset. Autonomous Exploration via Regions of Interest. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2003a, Las Vegas, USA, pp.1691-1696.

Grabowski, R., P. Khosla and H. Choset. An Enhanced Occupancy Map for Exploration via Pose Separation. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2003b, Las Vegas, USA, pp.705-710.

Guivant, J. and R. Katz. Global Urban Localization based on Road Maps. In Proc IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2007, San Diego, CA, USA, pp.1079-1084.

Hagen, S.H.G.T. Continuous State Space Q-Learning for Control of Nonlinear Systems. PhD Thesis, Computer Science Institute, University of Amsterdam. 2001.

Harmon, S.Y. The Ground Surveillance Robot (GSR): An Autonomous Vehicle Designed to Transit Unknown Terrain. IEEE Journal of Robotics and Automation, *3(3)*, pp.266-279. 1987.

He, T., C. Huang, B. Lum, J. Stankovic and T. Adelzaher. Range-free Localization Schemes for Large Scale Sensor Networks. In Proc. Annual International Conference on Mobile Computing and Networking, September 2003, San Diego, USA, pp.81-95.

Hoshino, M., H. Asama, K. Kawabata, Y. Kunii and I. Endo. Communication Learning for Cooperation among Autonomous Robots. In Proc. IEEE Conference on Industrial Electronics Society, October 2000, Nagoya, Japan, pp.2111-2116.

Howard, A., M.J. Mataric and G.S. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In Distributed Autonomous Robotic Systems, ed by H. Asama, T. Arai, T. Fukuda and T. Hasegawa, pp.299-308. Germany: Springer. 2002.

Huang, Y.F. and K. Gupta. An Adaptive Configuration-Space and Work-Space Based Criterion for View Planning. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, August 2005, Edmonton, Canada, pp.3366-3371.

Ichikawa, S., F. Hara and H. Hosokai. Cooperative Route-Searching Behavior of Multi-Robot System Using Hello-Call Communication. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, July 1993, Yokohama, Japan, pp.1149-1156.

Ikenoue, S., M. Asada and K. Hosoda. Cooperative Behavior Acquisition by Asynchronous Policy Renewal that Enables Simultaneous Learning in Multiagent Environment. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, September 2002, Lausanne, Switzerland, pp.2728-2734.

Ito, K. and S. Sakane. Robust View-based Visual Tracking with Detection of Occlusions. In Proc. IEEE International Conference on Robotics and Automation, May 2001, Seoul, Korea, pp.1207-1213.

Jouffe, L. Fuzzy Inference System Learning by Reinforcement Methods. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, *28(3)*, pp.338-355. 1998.

Jung, B. and G.S. Sukhatme. Cooperative Tracking Using Mobile Robots and Environment-Embedded, Networked Sensors. In Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation, August 2001, Banff, Canada, pp.206-211.

Jung, B. and G.S. Sukhatme. Tracking Targets Using Multiple Robots: The Effect of Environment Occlusion. Autonomous Robots, *13*, pp.191-205. 2002.

Kaelbling, L. P., M.L. Littman and A.W. Moore. Reinforcement Learning: A Survey. Artificial Intelligence Research, *4*, pp.237-285. 1996.

Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. Transaction of the ASME-Journal of Basic Engineering, *82*, pp.35-45. 1960.

Kawakami, K. I., K. Ohkura and K. Ueda. Adaptive Role Development in a Homogeneous Connected Robot Group. In Proc. IEEE International Conference on Systems, Man, and Cybernetics, October 1999, Tokyo, Japan, pp.254-256.

Khatib, O. and J.F. Le Maitre. Dynamic Control of Manipulators Operating in a Complex Environment. In Proc. 3rd CISM-IFToMM Symposium on Theory Practice Robots Manipulators, September 1978, Udine, Italy, pp.267-282.

Kim, J. H. and P. Vadakkepat. Multi-agent Systems: A Survey from the Robot-soccer Perspective. Intelligent Automation and Soft Computing, *6(1)*, pp.3-17. 2000.

Kleinrock, L., and J.A. Silvester. Optimum Transmission Radii for Packet Radio Networks or Why Six is a Magic Number. In Proc. IEEE National Telecommunications Conference, December 1978, Birmingham, USA, pp.4.3.1-4.3.5.

Ko, J., B. Stewart, D. Fox, K. Konolige and B. Limketkai. A Practical, Decision-Theoretic Approach to Multi-Robot Mapping and Exploration. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2003, Las Vegas, USA, pp.3232-3238.

Kobayashi, F., S. Sakai and F. Kojima. Determination of Exploration Target based on Belief Measure in Multi-Robot Exploration. In Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation, July 2003, Kobe, Japan, pp.1545-1550.

Koren, Y. and J. Borenstein. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In Proc. of IEEE International Conference on Robotics and Automation, April 1991, California, USA, pp.1398-1404.

Krishna, K.M. and H. Hexmoor. Social Control of a Group of Collaborating Multi-robot Multi-target Tracking Agents. In Proc. IEEE, AIAA 22nd Digital Avionics Systems Conference, October 2003, Indianapolis, USA, pp.8.C.2.1-8.C.2.7.

Kube, C.R. and H. Zhang. Stagnation Recovery Behaviors for Collective Robotics. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, September 1994, Munich, Germany, pp.1883-1890.

Kube, C.R. and H. Zhang. The Use of Perceptual Cues in Multi-Robot Box-Pushing. In Proc. IEEE International Conference of Robotics and Automation, April 1996, Minneapolis, USA, pp.2085-2090.

Kurabayashi, D. Toward Realization of Collective Intelligence and Emergent Robotics. In Proc. IEEE International Conference on Systems, Man, and Cybernetics, October 1999, Tokyo, Japan, pp.748-753.

Langendoen, K. and N. Reijers. Distributed Localization in Wireless Sensor Networks: a Quantitative Comparison. International Journal of Computer and Telecommunications Networking, *43(4)*, pp.499-518. 2003.

Levis, P., S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer and D. Culler. The Emergence of Networking Abstractions and Techniques in TinyOS. In Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation, March 2004, San Francisco, USA, pp.1-14.

Lim, J.G. and S.V. Rao. Mobility-enhanced Positioning in Ad Hoc Networks. In Proc. IEEE Wireless Communications and Networking Conference, March 2003, New Orleans, USA, pp.1832-1837.

Liu, Z., M.H. Ang and W.K.G. Seah. A Potential Field based Approach for Multi-Robot Tracking of Multiple Moving Targets. In Proc. 1st International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management, March 2003, Manila, Philippines.

Liu, Z., M.H. Ang and W.K.G. Seah. A Searching and Tracking Framework for Multi-Robot Observation of Multiple Moving Targets. International Journal of Advanced Computational Intelligence and Intelligent Informatics, *8(1)*, pp.14-22. 2004a.

Liu, Z., M.H. Ang and W.K.G. Seah. Searching and Tracking for Multi-Robot Observation of Moving Targets. In Proc. 8th Conference on Intelligent Autonomous Systems, March 2004b, Amsterdam, Netherlands, pp.157-164.

Liu, Z., M.H. Ang and W.K.G. Seah. Multi-Robot Concurrent Learning in Museum Problem. In Proc. 7th International Symposium on Distributed Autonomous Robotic Systems, June 2004c, Toulouse, France.

Liu, Z., M.H. Ang and W.K.G. Seah. Multi-Robot Concurrent Learning of Fuzzy Rules for Cooperation. In Proc. 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation, June 2005a, Espoo, Finland, pp.713-719.

Liu, Z., M.H. Ang and W.K.G. Seah. Reinforcement Learning of Cooperative Behaviors for Multi-Robot Tracking of Multiple Moving Targets. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, August 2005b, Edmonton, Canada, pp.1289-1294.

Liu, Z., M.H. Ang and W.K.G. Seah. Multi-Robot Concurrent Learning of Cooperative Behaviors for the Tracking of Multiple Moving Targets. Special issue on Computational Intelligence and Its Applications to Mobile Robots and Autonomous Systems, International Journal of Vehicle Autonomous Systems, *4(2/3/4)*, pp.196-215, 2006.

Longman. Longman Dictionary of Contemporary English, International Students Edition. Edinburgh Gate Harlow, England: Pearson Education Limited. 1995.

Low, K.H. Integrated Robot Planning and Control with Extended Kohonen Maps. Master Thesis, Department of Computer Science, School of Computing, National University of Singapore. 2002.

Low, K.H., W.K. Leow and M.H. Ang. Enhancing the Reactive Capabilities of Integrated Planning and Control with Cooperative Extended Kohonen Maps. In Proc. IEEE International Conference on Robotics and Automation, September 2003, Taipei, Taiwan, pp.3428-3433.

Mataric, M.J. Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents. Journal of Cognitive Systems Research, *2(1)*, pp.81-93. 2001.

Mattos, L. and E. Grant. Passive Sonar Applications: Target Tracking and Navigation of An Autonomous Robot. In Proc. IEEE International Conference on Robotics and Automation, April 2004, New Orleans, USA, pp.4265-4270.

Metropolis, N. and S. Ulam. The Monte Carlo Method. Journal of the American Statistical Association, *44*, pp.335-341. 1949.

Miyata, N., J. Ota, T. Arai and H. Asama. Cooperative Transport by Multiple Mobile Robots in Unknown Static Environments Associated with Real-Time Task Assignment. IEEE Transactions on Robotics and Automation, *18(5)*, pp.769-780. 2002.

Molnar, P. and J. Starke. Control of Distributed Autonomous Robotic Systems Using Principles of Pattern Formation in Nature and Pedestrian Behavior. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, *31(3)*, pp.433-435. 2001

Moorehead, S.J., R. Simmons and W.L. Whittaker. Autonomous Exploration Using Multiple Sources of Information. In Proc. IEEE International Conference on Robotics and Automation, May 2001, Seoul, Korea, pp.3098-3103.

Moorehead, S.J. Autonomous Exploration of Cliff Faces Using Multiple Information Metrics. In Proc. IEEE International Conference on Robotics and Automation, May 2002, Washington, USA, pp.2668-2673.

Muppirala, T., S. Hutchinson and R. Murrieta-Cid. Optimal Motion Strategies Based on Critical Events to Maintain Visibility of a Moving Target. In Proc. IEEE International Conference on Robotics and Automation, April 2005, Barcelona, Spain, pp.3826-3831.

Murphy, R.R., C.L. Lisetti, R. Tardif, L. Irish and A. Gage. Emotion-Based Control of Cooperating Heterogeneous Mobile Robots. IEEE Transactions on Robotics and Automation, *18(5)*, pp.744-757. 2002.

Murrieta, R., A. Sarmiento, S. Bhattacharya and S.A. Hutchinson. Maintaining Visibility of a Moving Target at a Fixed Distance: the Case of Observer Bounded Speed. In Proc. IEEE International Conference on Robotics and Automation, April 2004, New Orleans, USA, pp.479-484.

Nagatani, K. and H. Choset. Toward Robust Sensor based Exploration by Constructing Reduced Generalized Voronoi Graph. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 1999, Kyongju, Korea, pp.1687-1692.

Newman, P., M. Bosse and J. Leonard. Autonomous Feature-based Exploration. In Proc. IEEE International Conference on Robotics and Automation, September 2003, Taipei, Taiwan, pp.1234-1240.

Niculescu, D. and B. Nath. Ad hoc Positioning System (APS). In Proc. IEEE Global Communications Conference, November 2001, San Antonio, USA, pp.2926-2931.

Niculescu, D. and B. Nath. DV Based Positioning in Ad Hoc Networks. Journal of Telecommunication Systems, *22(1-4)*, pp.267-280. 2003.

Ogras, U.Y., O.H. Dagci and U. Ozguner. Cooperative Control of Mobile Robots for Target Search. In Proc. IEEE International Conference on Mechatronics, June 2004, Istanbul, Turkey, pp.123-128.

Ohkawa, K., T. Shibata and K. Tanie. Method for Generating of Global Cooperation Based on Local Communication. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 1998, Victoria, Canada, pp.108-113.

Olson, C.F. Probabilistic Self-Localization for Mobile Robots. IEEE Transactions on Robotics and Automation, *16(1)*, pp.55-66. 2000.

Parker, L.E. Multi-Robot Team Design for Real-World Applications. In Distributed Autonomous Robotic Systems 2, ed by H. Asama, T. Fukuda, T. Arai and I. Endo, pp.91-102. Tokyo: Springer-Verlag. 1996.

Parker, L.E. Cooperative Motion Control for Multi-target Observation. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, September 1997, Grenoble, France, pp.1591-1597.

Parker, L.E. ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation. IEEE Transactions on Robotics and Automation, *14(2)*, pp.220-240. 1998.

Parker, L.E. Distributed Algorithm for Multi-Robot Observation of Multiple Moving Targets. Autonomous Robots, *12(3)*, pp.231-255. 2002.

Perkins, C.E. and E.M. Royer. Ad hoc On-Demand Distance Vector Routing. In Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications, February 1999, New Orleans, USA, pp.90-100.

Pirjanian, P. and M.J. Mataric. Multi-Robot Target Acquisition Using Multiple Objective Behavior Coordination. In Proc. IEEE International Conference on Robotics and Automation, April 2000, San Francisco, USA, pp.2696-2702.

Rao, N.S.V. and S.S. Iyengar. Autonomous Robot Navigation in Unknown Terrains: Incidental Learning and Environmental Exploration. IEEE Transactions on Systems, Man and Cybernetics, *20(6)*, pp.1443-1449. 1990.

Reynaud, E. and D. Puzenat. A Multisensory Identification System for Robotics. In Proc. International Joint Conference on Neural Networks, July 2001, Washington, USA, pp.2924-2929.

Rogge, J. and D. Aeyels. Sensor Coverage with a Multi-Robot System. In Proc. IEEE International Symposium on Intelligent Control, October 2007, Singapore, pp.71-76.

Roumeliotis, S.I. and I.M. Rekleitis. Analysis of Multirobot Localization Uncertainty Propagation. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2003, Las Vegas, USA, pp.1763-1770.

Savarese, C., J. Rabaey and K. Langendoen. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In Proc. USENIX Annual Technical Conference, June 2002, Monterey, Canada, pp.317-327.

Schulz, D., W. Burgard, D. Fox and A.B. Cremers. Tracking Multiple Moving Targets with a Mobile Robot Using Particle Filters and Statistical Data Association. In Proc. IEEE International Conference on Robotics and Automation, May 2001, Seoul, Korea, pp.1665-1670.

Seah, W.K.G., Z. Liu, J.G. Lim, S.V. Rao and M.H. Ang. TARANTULAS: Mobility-enhanced Wireless Sensor-Actuator Networks. In Proc. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, June 2006, Taichung, Taiwan, pp.548-551.

Seah, W.K.G., H.X. Tan, Z. Liu and M.H. Ang. Multiple-UUV Approach for Enhancing Connectivity in Underwater Ad-hoc Sensor Networks. In Proc. MTS/IEEE OCEANS Conference, September 2005, Washington, USA, pp.2263-2268.

Seah, W.K.G. and H.X. Tan. Multipath Virtual Sink Architecture for Underwater Sensor Networks. In Proc. MTS/IEEE OCEANS Asia Pacific Conference, May 2006, Singapore.

Singer, R. and R. Sea. New Results in Optimizing Surveillance System Tracking and Data Correlation Performance in Dense Multitarget Environments. IEEE Transactions on Automatic Control, *18(6)*, pp.571-582. 1973.

Skrzypczyk, K. Game Theory based Target Following by a Team of Robots. In Proc. 4th International Workshop on Robot Motion and Control, June 2004, Puszczykowo, Poland, pp.91-96.

Smart, W.D. and L.P. Kaelbling. Practical Reinforcement Learning in Continuous Spaces. In Proc. 17th International Conference on Machine Learning, June 2000, Stanford, USA, pp.903-910.

Sit, T.C.H, Z. Liu, M.H. Ang and W.K.G. Seah. Multi-Robot Mobility Enhanced Hop-Count Based Localization in Ad Hoc Networks. Journal of Robotics and Autonomous Systems, *55(3)*, pp.244-252, March 2007.

Stone, P. RoboCup as an Introduction to CS Research. In RoboCup-2003: Robot Soccer World Cup VII, ed by D. Polani, B. Browning, A. Bonarini and K. Yoshida, pp.284-295. Berlin: Springer Verlag. 2003.

Sugiyama, H., T. Tsujioka, and M. Murata. Coordination of Rescue Robots for Real-Time Exploration Over Disaster Areas. In Proc IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC), May 2008, Orlando, Florida, USA, pp.170-177.

Sutton, R.S. and A.G. Barto. Reinforcement Learning: An Introduction. Cam-bridge, MA: MIT Press. 1998.

Suzuki, I. and M. Yamashita. Searching for a Mobile Intruder in a Polygonal Region. SIAM Journal on Computing, *21(5)*, pp.863-888. 1992.

Tangamchit, P., J.M. Dolan and P.K. Khosla. The Necessity of Average Rewards in Cooperative Multirobot Learning. In Proc. IEEE International Conference on Robotics and Automation, May 2002, Washington, USA, pp.1296-1301.

Thrun, S. Robotic Mapping: A Survey. Technical Report: CMU-CS-02-111, Carnegie Mellon University. 2002.

Trevai, C., F. Fukazawa, H. Yuasa, J. Ota, T. Arai and H. Asama. Exploration-Path Generation of Multiple Mobile Robots Using Reaction-Diffusion Equation on a Graph. In Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation, July 2003, Kobe, Japan, pp.1114-1119.

Uchibe, E., M. Nakamura and M. Asada. Co-evolution for Cooperative Behavior Acquisition in A Multiple Mobile Robot Environment. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, October 1998, Victoria, Canada, pp.425-430.

Olivier, M. Webots: Professional Mobile Robot Simulation. International Journal of Advanced Robotic Systems, *1(1)*, pp.39-42. 2004.

Werger, B.B. and M.J. Mataric. Broadcast of Local Eligibility: Behavior-Based Control for Strongly-Cooperative Robot Teams. In Proc. 4th International Conference on Autonomous Agents, June 2000, Barcelona, Spain, pp.21-22.

Wong, S.Y., J.G. Lim, S.V. Rao and W.K.G. Seah. Hopcount Localization with Density and Path Length Awareness in Non-Uniform Wireless Sensor Networks. In Proc. IEEE 61st Vehicular Technology Conference, May 2005, Stockholm, Sweden, pp.2551-2555.

Wullschleger, F.H., K.O. Arras and S.J. Vestli. A Flexible Exploration Framework for Map Building. In Proc. 3rd European Workshop on Advanced Mobile Robots, September 1999, Zurich, Switzerland, pp.49-56.

Yamashita, A., T. Arai, J. Ota and H. Asama. Motion Planning of Multiple Mobile Robots for Cooperative Manipulation and Transportation. IEEE Transactions on Robotics and Automation, *19(2)*, pp.223-237. 2003.

Yamashita, M., H. Umemoto, I. Suzuki and T. Kameda. Searching for Mobile Intruders in a Polygonal Region by a Group of Mobile Searchers. Algorithmica, *31*, pp.208-236. 2001.

Yamauchi, B. A Frontier based Approach for Autonomous Exploration. In Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation, July 1997, Monterey, Canada, pp.146-151.

Yamauchi, B. Frontier-based Exploration using Multiple Robots. In Proc. of 2nd International Conference on Autonomous Agents, May 1998, Minneapolis, USA, pp.47-53.

Yamauchi, B., A. Schultz and W. Adams. Integrating Exploration and Localization for Mobile Robots. Adaptive Behavior, *7(2)*, pp.217-229. 1999.

Yan, X.W., Z.D. Deng and Z.Q. Sun. Genetic Takagi-Sugeno Fuzzy Reinforcement Learning. In Proc. IEEE International Symposium on Intelligent Control, September 2001, Mexico City, Mexico, pp.67-72.

Ye, C., N.H.C. Yung and D. Wang. A Fuzzy Controller with Supervised Learning assisted Reinforcement Learning Algorithm for Obstacle Avoidance. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, *33(1)*, pp.17-27. 2003.

Zeng, X., R. Bagrodia and M. Gerla. GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks. In Proc. 12th Workshop on Parallel and Distributed Simulations, May 1998, Banff, Canada, pp.154-161.

Zhang, F., W.D. Chen and Y.G. Xi. Improving Collaboration through Fusion of Bid Information for Market-based Multi-robot Exploration. In Proc. IEEE International Conference on Robotics and Automation, April 2005, Barcelona, Spain, pp.1157-1162.

Zlot, R. and A. Stentz. Market-based Multirobot Coordination for Complex Tasks. International Journal of Robotics Research, *25(1)*, pp.73-101. 2006.